NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMAL PATH PLANNING FOR MULTI-ARM, MULTI-LINK ROBOTIC MANIPULATORS

by

Joseph A. Cascio

December 2008

Thesis Advisor:	I. M. Ross
Second Reader:	A. D. Scott

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2008	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Optimal Path Planning for Multi-Arm, Multi-Link Robotic Manipulators	5. FUNDING NUMBERS
6. AUTHOR(S) Joseph A. Cascio	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

This work investigates the problem of robotic arm control with the goal of achieving given performance requirements by solving for the optimal joint trajectories and corresponding controls for tasks, such as point-to-point positioning. The resulting optimal control problem is highly nonlinear and constrained due to the nonlinearities in the robotic arm dynamics and kinodynamic constraints including limits on joint velocities and actuator torques.

This thesis illustrates the applicability of pseudospectral methods to solve the optimal path planning problem for a system of multi-link, multi-degree of freedom robotic arms. The optimal control problem is defined in standard form and solved using the software package DIDO. Pontryagin's Minimum Principle is used to verify that the proposed solution satisfies the necessary conditions for optimality. A particularly challenging aspect that is explored is the optimal motion of multiple arms conducting independent tasks with the risk of collision. Collision avoidance can be achieved by modeling appropriate path constraints.

The processes for optimal trajectory planning are developed for a single two degree-of-freedom manipulator conducting point-to-point positioning and extended to include dual three degree-of-freedom manipulator maneuvers employing collision avoidance. The results demonstrate the suitability of pseudospectral techniques to solving the minimum time and minimum control maneuvers for robotic arms. The employment of collision avoidance techniques will facilitate continued research in autonomous robotic motion planning using optimal control criteria in multiple arm systems.

14. SUBJECT TERMS Optimal Control, Trajectory Optimization, Path Planning, Robotic Manipulator, Collision Avoidance, DIDO, Pseudospectral Methods, Autonomous Robotic Control	15. NUMBER OF PAGES 121
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

OPTIMAL PATH PLANNING FOR MULTI-ARM, MULTI-LINK ROBOTIC MANIPULATORS

Joseph A. Cascio
Lieutenant Commander, United States Navy
B.S., University of Southern California, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2008

Author: Joseph A. Cascio

Approved by: I. M. Ross
Thesis Advisor

A. D. Scott
Second Reader

K. Millsaps
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This work investigates the problem of robotic arm control with the goal of achieving given performance requirements by solving for the optimal joint trajectories and corresponding controls for tasks, such as point-to-point positioning. The resulting optimal control problem is highly nonlinear and constrained due to the nonlinearities in the robotic arm dynamics and kinodynamic constraints including limits on joint velocities and actuator torques.

This thesis illustrates the applicability of pseudospectral methods to solve the optimal path planning problem for a system of multi-link, multi-degree of freedom robotic arms. The optimal control problem is defined in standard form and solved using the software package DIDO. Pontryagin's Minimum Principle is used to verify that the proposed solution satisfies the necessary conditions for optimality. A particularly challenging aspect that is explored is the optimal motion of multiple arms conducting independent tasks with the risk of collision. Collision avoidance can be achieved by modeling appropriate path constraints.

The processes for optimal trajectory planning are developed for a single two degree-of-freedom manipulator conducting point-to-point positioning and extended to include dual three degree-of-freedom manipulator maneuvers employing collision avoidance. The results demonstrate the suitability of pseudospectral techniques to solving the minimum time and minimum control maneuvers for robotic arms. The employment of collision avoidance techniques will facilitate continued research in autonomous robotic motion planning using optimal control criteria in multiple arm systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

About a year ago, Professor Michael Ross presented the idea of formulating an optimal control problem that simultaneously solved the trajectories of multiple robotic arms without colliding. I of course had no background in robotics and some background in dynamics, but fresh out of my optimization courses, my initial reaction was “How hard could that be? Sign me up!” No sooner had I uttered those words did I realize how wrong I could be. Thank you Dr. Ross for allowing me the opportunity to think through the numerous road blocks of problem formulation.

However, it was the individuals in the Guidance, Navigation, and Control Lab at Naval Postgraduate School whose help allowed me to complete this thesis. Dr. Pooya Sekhavit provided countless hours of helping me debug my DIDO code and suggesting ways of improving its performance. Dr. Mark Karpenko provided background in robotics and computers that I lacked and formulated the dynamics for the arms. Dr. Qi Gong always managed to have the answer to the “hard question” and it was his insights on problem formulation led to the parameter optimization that is the heart of the obstacle avoidance methods I discuss.

I could not have asked for better classmates in the Space Systems Engineering curriculum. I learned more over the past 27 months from each of them than any of my courses. I will not forget you.

And of course my wife Karin and my beautiful daughters Bella, Amy, and Lily. Thank you for supporting me as I finished my coursework and thesis at NPS. Daddy is looking forward to being able to play with you again.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The use of robotic arms has become commonplace in today's technological society. These machines accomplish tasks from the mundane to the miraculous with little regard to how they calculate their motion by the majority of the populace, after all the motion control of one's arm is much more complicated than a simple robotic arm yet it is accomplished with little effort. The reality is that the mathematical model for robotic motion is highly nonlinear and not intuitive.

Optimal path planning is the primary means to achieve efficient control of robotic manipulators in physical space. Methods of achieving optimal control have been researched extensively over the past half century, but it is only with the increase in computational power and advanced numerical techniques over the past decade or so that optimal control can now be realized in a wide variety of tasks. The field of robotics has historically used some form of optimization to achieve efficiencies in the motion of systems with the goal of autonomous operations implied in many endeavors.

Terrestrial applications of robots abound and the economic benefit for achieving efficiencies in motion cannot be taken lightly and is a driving force behind many of the previous studies in robotic optimal control [1,2,3]. Even a small increase in efficiency can have a large effect on throughput. Some of these applications require a form of autonomous or adaptive path planning, and the vast majority of them are very specific tasks that are suited to specific methods of calculating highly accurate trajectories that satisfy specific conditions. Space applications of robotic manipulators also lend themselves towards exploring autonomous path planning for a variety of machines. Robot arms have been used extensively on the Space Shuttle and the International Space Station (ISS) with increasing levels of complexity. A very good example of the use of

cooperative robotic arms was in April 2001 when the ISS's Canadarm2 was used to transfer cargo directly with the shuttle Endeavor's arm (Figure 1, [4]). While most of this work was accomplished via manual control, only a little imagination is necessary to see the usefulness of autonomous control for a system of cooperating robotic manipulators. The Naval Research Laboratory (NRL) has been conducting research in participation with the Defense Advanced Research Project Agency (DARPA) on a program to use a system of robotic arms on an advanced serving satellite [5]. SUMO, or Spacecraft for the Universal Modification of Orbits, is a program that is being investigated to integrate autonomous rendezvous and grappling technologies into a relatively low cost means of altering a satellite's orbit. As Figure 2 shows, SUMO is envisioned to have three seven degree-of-freedom (DOF) robotic arms that will require advanced motion planning in a cluttered environment to operate safely. The NRL is developing control algorithms that operate in real time and must satisfy the non-linear dynamics and constraints of the system. Traditional optimal control techniques in the presence of obstacles were deemed "unsuitable for real-time" use due to their speed and complexity.

Figure 1. Cooperating Robotic Arms in Space: International Space Station Arm Canadarm2 Transfers Cargo to Endeavor's Arm April 28, 2001 (From [4])

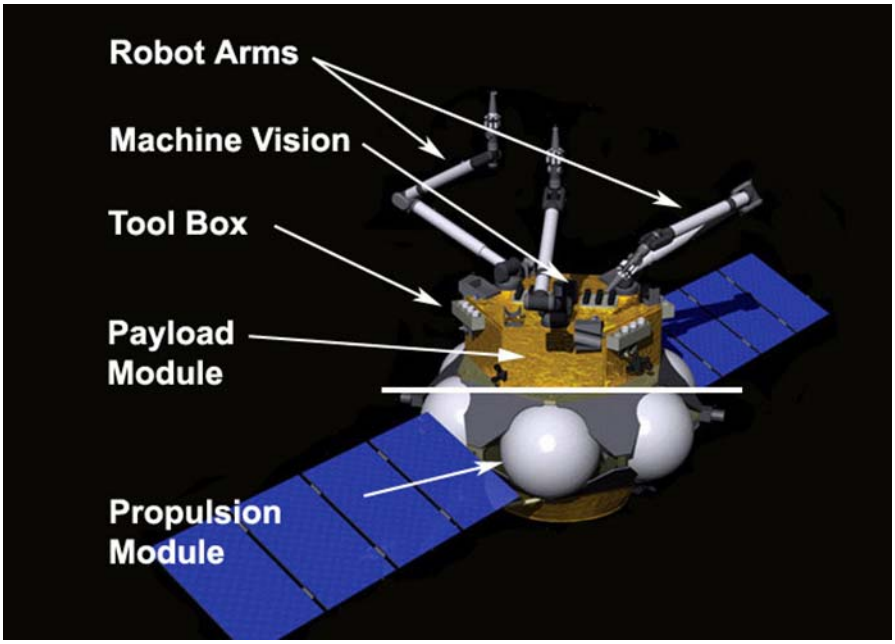

Figure 2. SUMO Advanced Servicing Spacecraft (From [5])

B. TRAJECTORY PLANNING

Traditionally, the study of robotic control has been approached in two stages or levels (see Figure 3). The first stage is off-line path planning or trajectory planning. Various methods are utilized to compute a suitable state trajectory for the robot to achieve the desired task given the physical constraints on the system. Once that path has been defined, an inner loop controller can be used in real time to correct the errors in the path of the robot's actual trajectory within the tolerance value of the desired path [6,7,8]. Most previous work has been focused on the path planning problem and defining some form of an optimal, collision free trajectory. Ideally, the computation of the optimal control trajectory could take place within a controller and allow a single stage control of robotic systems. However, computation time required for the path planning problem typically makes this solution infeasible to implement in real time. Figure 3 diagrams a number of robotic control methods and illustrates the two-level approach described [3, 8].

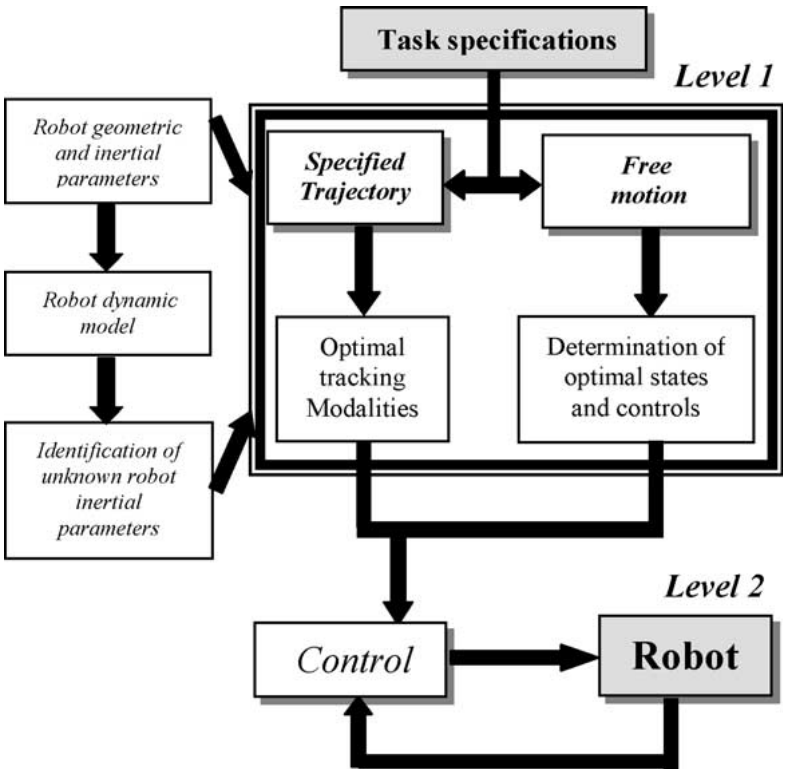

Figure 3. Synoptic Diagram of Optimal Motion Planner for Robotic Manipulators (From [8]).

The problem of trajectory planning can be formulated as a general optimal control problem that seeks to minimize some cost function (also called an objective function in some literature). Given the state variables $\mathbf{x} \in \mathbb{R}^{N_X}$ and $\mathbf{u} \in \mathbb{R}^{N_U}$, the standard optimal control problem is written in the form:

$$\begin{aligned} \text{Minimize} \quad & J\left[\mathbf{x}(\cdot),\mathbf{u}(\cdot),t_f\right] = E\left(x(t_f),t_f\right) + \int_{t_0}^{t_f} F\left(\mathbf{x}(\cdot),\mathbf{u}(\cdot),t\right)\mathrm{dt} \\ \text{Subect to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t),\mathbf{u}(t),t\right) \\ & \mathbf{x}(t_0) = \mathbf{x}^0 \\ & \mathbf{e}^L(t_f) \le \mathbf{e}\left(\mathbf{x}(t_f),t_f\right) \le \mathbf{e}^U(t_f) \\ & \mathbf{h}^L(t) \le \mathbf{h}\left(\mathbf{x}(\cdot),\mathbf{u}(\cdot),t\right) \le \mathbf{h}^U(t) \end{aligned} \tag{1}$$

where J is the cost function, E is the endpoint cost, F is the running cost, $\mathbf{f}$ is the set of equations describing the dynamics of the system, $\mathbf{e}$ is the set of endpoint

constraints, and **h** is the set of path constraints [9]. The most common optimization problem in robotics is to solve the minimum time problem where the cost function $E=t_f$. A running cost such as $F=\boldsymbol{u}^T\boldsymbol{u}$ is also used to reduce the control effort. Variations on this basic cost function have been explored in the literature [2,8,10,11,12,13]. The endpoint constraints define the final state of the system. The path constraints can include torque and velocity limits on the joints of the manipulator, geometric limits to the system, criteria for obstacle avoidance, and path tracking depending on the task. A trajectory is considered feasible, though not necessarily optimal, if the dynamics, endpoint constraints, and path constraints are satisfied.

Trajectory planning for robotics typically takes one of two forms: a decoupled approach, which first finds a feasible path and then optimizes the control along that path; and a direct trajectory planning approach, which includes taking kinodynamic constraints of the system to solve an optimal path [7, 8]. These two methods each have merits and drawbacks.

A substantial amount of research has been focused on decoupled methods by improving the performance of a robotic system given a specified collision-free path [6,7,8,13,14,15]. These types of trajectories are particularly useful in tasks that require specific trajectories to accomplish the assigned task. As an example, [13] presents an algorithm that searches for a time optimal control trajectory while minimizing the control effort along a specified path. This formulation is fairly typical of previous methods and useful as a brief illustration. For a robotic manipulator, the specified path is normally defined by the position and orientation of the tool being used. The dynamics of the system can be derived using a number of methods such as the Lagrange–Euler method. The path is first specified in terms of the generalized joint coordinates and then written in parametric form using a standard interpolation method. The method of parameterizing the motion of the robot is complex and not trivial. The path is scaled and its first, second, and third order derivatives are calculated. The cost function is then restated in terms of the scaled parametric values and minimized

in terms of the parametric variables, which are compared to the kinematic constraints of the system to check for feasibility. Reference [13] uses a genetic algorithm to search for the optimal solution, while reference [10] proposes phase plane techniques that solve the minimum time problem along a desired path given in parametric form. Such methods often assume some form of bang-bang control near the limits of the feasible joint space, which may not be desirable since the sudden change in manipulator torques can cause damage, or excessive wear, or excite flexible modes.

When the path is not specified, approaches have been proposed to solve the global time optimal solution using grid or cell type iterations based on the decoupled path planning problem method [16, 17]. These are highly involved and can only be considered to converge to near-optimal trajectories in some cases [6]. In general, these approaches attempt to optimize every possible path and are therefore not practical in high dimensional problems.

Robotic manipulators usually consist of multiple links and can achieve an objective maneuver through multiple paths. Optimization results often produce surprising results that can be physically explained, but may differ from an intuitive solution [3]. Full motion planning or direct trajectory planning computes the optimal solution in the state space of the system and solves an unknown optimal trajectory of each joint. When the movement of the robot is not specified, these trajectory methods are more useful than the decoupled path planning algorithms at minimizing some cost function over the semi-infinite range of possible trajectories. Many direct trajectory planning techniques have been explored over the past three decades. Excellent surveys can be found in [6], [8], and [19]. Figure 4 is a useful diagram that summarizes some of the methods used to solve the trajectory planning problems. These algorithms can be used to solve point-to-point, or pick-and-place trajectory problems where only the initial and final positions of the end effector are defined. Path constrained optimization problems can also be solved. Methods of computing optimal trajectories generally fall within two categories: direct and indirect [8].

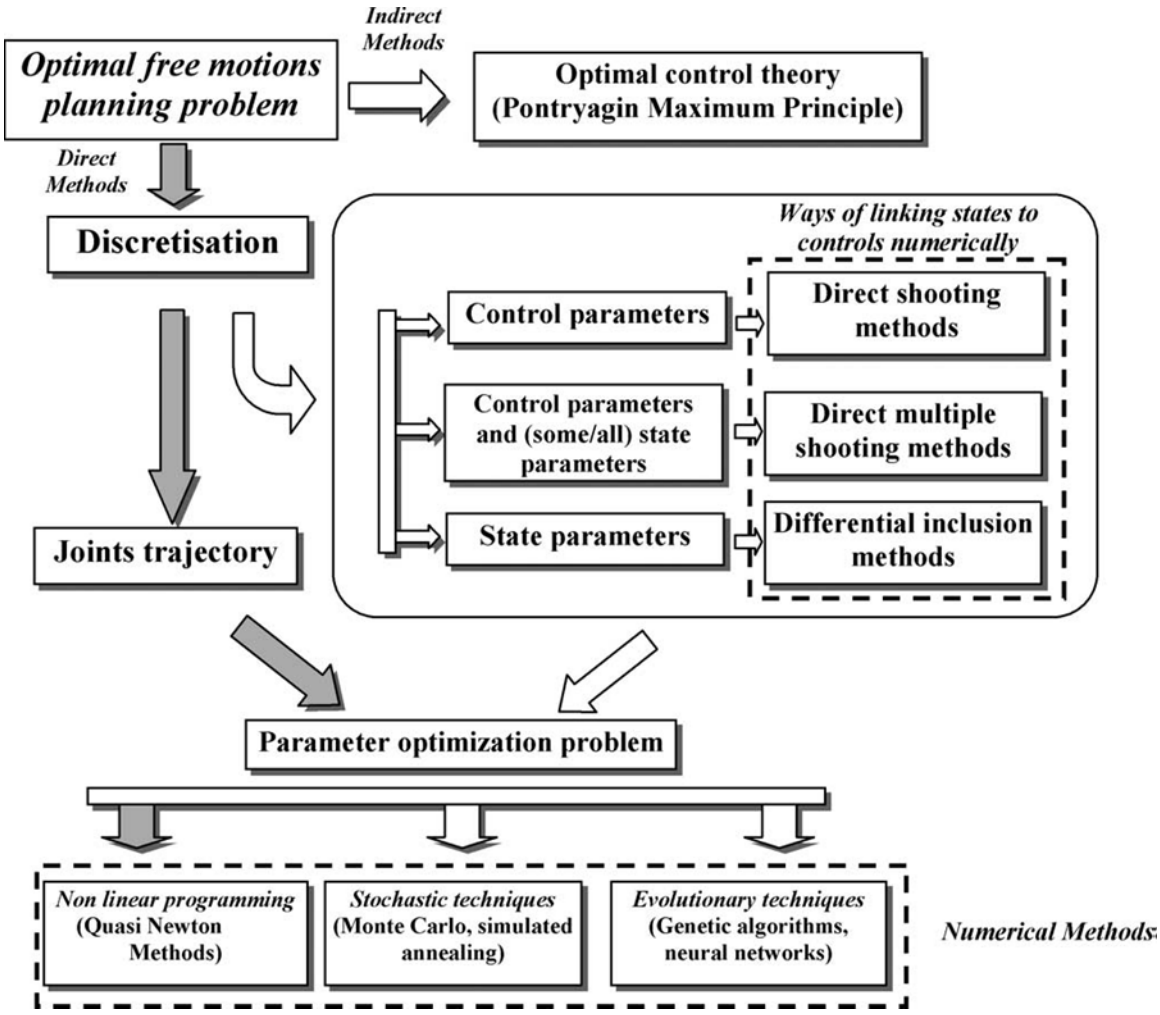

Figure 4. Optimal Free Trajectory Planning Problem Methods (From [8])

Indirect methods are primarily based on Pontryagin's Minimum Principle and solving the necessary conditions for optimality [18,19]. Given a standard problem formulation of the form of Equation (1), the control Hamiltonian is defined where $H \in \square$ and $\boldsymbol{\lambda} \in \square^{N_x}$ is the adjoint covector or costate vector:

$$H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := F(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{2}$$

Then applying the Lagrange multiplier, $\boldsymbol{\mu} \in \square^{N_\mathbf{h}}$ in order to define the Lagrangian of the Hamiltonian yields:

$$\bar{H}(\boldsymbol{\mu}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) = H + \boldsymbol{\mu}^T \mathbf{h} \tag{3}$$

Similarly, an end-point Lagrangian is given by

$$\bar{E} = E + \mathbf{v}^T \mathbf{e} \tag{4}$$

where $\mathbf{v} \in \mathbb{R}^{N_\mathbf{e}}$. The lower Hamiltonian, $\mathscr{H}$ is the value of the Hamiltonian with the optimum control profile and is defined as

$$\mathscr{H}(\boldsymbol{\lambda}, \mathbf{x}, t) = \min_{\mathbf{u} \in \mathbb{R}^{N_U}} H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) \tag{5}$$

The necessary conditions can then be described by Equations (6)-(10) below [9].

Hamiltonian Minimization Condition: $$\frac{\partial \bar{H}}{\partial \mathbf{u}} = 0 \tag{6}$$

Adjoint Condition: $$-\dot{\boldsymbol{\lambda}} = \frac{\partial \bar{H}}{\partial \mathbf{x}} \tag{7}$$

Transversality Condition: $$\boldsymbol{\lambda}(t_0) = -\frac{\partial \bar{E}}{\partial \mathbf{x}(t_0)} \qquad \boldsymbol{\lambda}(t_f) = \frac{\partial \bar{E}}{\partial \mathbf{x}(t_f)} \tag{8}$$

Hamiltonian Value Condition: $$\bar{H}(t_0) = \frac{\partial \bar{E}}{\partial t_0} \qquad \bar{H}(t_f) = -\frac{\partial \bar{E}}{\partial t_f} \tag{9}$$

Hamiltonian Evolution Condition: $$\dot{\mathscr{H}} = \frac{\partial \bar{H}}{\partial t} \tag{10}$$

The optimal trajectory can then be found by solving the multi-point boundary value problem by a multiple shooting method though convergence to a solution can be difficult to predict [8]. An excellent survey of indirect methods for computing the optimal trajectory for robotic manipulators is presented in [19]. The penalty for applying indirect methods to solve the optimal control problem is the amount of effort required to deriving the necessary conditions, particularly for complex systems.

The most common method to solving optimal control problems in recent years is by using direct methods [6,8,15,19]. In general these methods depend on discretizing the state and control variables and using nonlinear optimization techniques [15], evolutionary or hierarchical approaches [20,21], or stochastic methods to optimize the solution. The latter two techniques tend to suffer from "numerical explosion when treating high dimension problems [8]."

Almost all the methods to solve optimal problems require a level of discretizing the state and control parameters in some way. Traditional parameterization is done by uniformly distributing nodes along the time profile and solving a constrained optimization problem using gradient based nonlinear optimization techniques such as sequential quadratic programming (SQP). An in depth approach to treating the optimal control problem via a direct means is presented in [3] and is representative of nonlinear optimization techniques. An initial control function is used to provide an initial guess to the algorithm within the feasible bounds of **u**. The corresponding state values, **x**, are then approximated using a collocation technique that results in a piece-wise trajectory that satisfies the system dynamics and constraints via a multiple shooting method. The state and control function can then be parameterized and used to define the cost function. This results in a large scale nonlinear optimization problem. A numerical algorithm, such as SQP, which is tailored to the structure of the problem, is then used.

Regardless of the techniques used to solve the path planning problem, the direct and indirect methods can be used to complement each other. A candidate solution can be verified by comparing the results to the necessary conditions for optimality [22]. A hybrid solution is presented in [2] that uses a direct collocation approach, which parameterizes the state, and control variables with a poor initial guess trajectory using the endpoints to interpolate a discreet solution. This is then used to approximate a nonlinear optimization problem using SQP. Pontryagin's necessary conditions are then calculated symbolically based on the dynamics and constraints of the system. The results of the direct collocation

calculation can then be used as the initial guess to solve the multipoint boundary value problem. While this method is useful, the calculation of the necessary conditions are computationally intensive and make solving anything more complex than a 3-DOF system difficult [2].

Pseudospectral methods have also been proposed to solve the trajectory planning problem for robotic manipulators [23]. This method differs from previous direct methods primarily by the discretization method. The Legendre pseudospectral method approximates the state and control variables using particular interpolating polynomials. The discretized nodes are non-uniformly spaced based on Legendre-Gauss-Lobatto point allocation. The Covector Mapping Theorem allows this method to "make no distinction between the so-called direct and indirect methods" [22] and lends itself to verification by application of Pontryagin's Minimum Principle with the automatic generation of states, costates, and other dual variables.

C. THE COOPERATIVE PLANNING PROBLEM

Multiple robots working in the same space complicate the path-planning problem exponentially. There are generally two traditional approaches to attacking the problem of coordinated movement between multiple robots: centralized planning and decoupled planning [6]. The latter consists of finding feasible paths for each robot without regard to the other robots, in other words disregarding any risk of collision. The robots are then sequenced by adjusting start times and velocities of the individual arms to avoid potential collisions [24,25]. This method may be considered attractive from a computational aspect because the dimensions of the problem are limited to a single robot at a time; however, the system is not optimized. The former robots become solid obstacles in the way of the latter's motion and may shrink their feasible work space to zero. Cooperation between the robots is unlikely.

Centralized planning computes the trajectory of all elements of the system simultaneously and effectively treats the system as a single state vector. As the

number of individual robots increase, the dimension of the problem increases accordingly. When the centralized planning problem is solved, the solution provides the state and control trajectory for the complete system. While computation speed may seem a concern in implementing the method on practical applications, increased computing power and efficient numerical techniques are allowing centralized planning of more and more complex systems to be plausible.

Obstacle avoidance algorithms rely on measuring the distance between the objects and including some buffer. A number of methods that maneuver a point target around an obstacle-rich environment have been previously proposed [26,27]. Distance functions are used in most algorithms that use geometric shapes to define the obstacles, most of which compare the set of points for one obstacle to the set of points in the other obstacle and finding the minimum distance. This requires some means of discretizing the obstacles and using a brute force method to compare each point [6,28]. For example using a finite number of different radii spheres to cover the obstacles and constrain the minimum distance between spheres [8]. This formulation would define a path constraint such that at each point in time the minimum distance between the centers of each sphere is greater than some reference distance.

A variation of this technique was presented in [29] where a distance function from a set of reference points of each manipulator to the obstacle was computed.

$$D_k\left(M_k^l, P_k^i\right) = \left(\frac{[M_k^l]_x - [P_k^i]_x}{L_x}\right)^2 + \left(\frac{[M_k^l]_y - [P_k^i]_y}{L_y}\right)^2 + \left(\frac{[M_k^l]_z - [P_k^i]_z}{L_z}\right)^2$$

where D_k is the distance, M_k^l is a set of l= 1,…,n points that describe the contour of the manipulator arms at time k, P_k^i is a set of points that describe the center of each obstacle, i, at time k, and L_j describes the obstacle size in each direction. The cost function can then include the term J_D where J_D^0 is a reference distance:

$$J_D = \frac{-\max\left[\min_{k=0\ldots N} D_k\right]}{J_D^0}$$

A collision, in theory, would make J_D=0 which is the maximum value in this formulation. A collision free path, assuming the function is properly scaled, would have a lower cost than a path with a collision. However, there is no guarantee that the system will avoid a collision.

D. A COOPERATIVE CONTROL PROPOSAL

Optimal cooperative path planning consists of simultaneously solving trajectories for multiple robotic arms while meeting all obstacle avoidance criteria. Rather than discretizing the links of an arm and solving the avoidance problem in the work space, this thesis proposes reformulating the minimum distance problem in a parametric form and solving a parameterized optimization problem. Pseudospectral optimal control formulations readily lend itself to solving such problems.

Using the software program DIDO, pseudospectral methods are used to solve the optimal minimum-time trajectory for 2-DOF and 3-DOF robotic manipulators conducting point-to-point maneuvers where the initial point is given in joint coordinates and the final point in Cartesian coordinates. The computed state, costate, and control trajectories facilitate verifying Pontryagin's necessary conditions for optimality. The calculated control trajectory is then propagated to demonstrate feasibility. Building on these results, dual manipulator solutions are analyzed and presented using variations of cost functions. Finally, two methods of formulating obstacle avoidance criteria are presented and compared in various scenarios. A static obstacle avoidance formulation is first illustrated followed by a system of two multiple link arms performing a simple pick-and-place operation.

II. OPTIMAL TRAJECTORY PLANNING

Pseudospectral (PS) methods have been used to solve a variety of trajectory planning and optimal control problems as shown in [27,30,31,32,33]. PS methods have been flight tested onboard the International Space Station, and other flight experiments are in the planning stages. Given their widespread applicability, it should be no surprise that the manipulator path planning problem is a logical application of these techniques. Motion planning for time-optimal point-to-point maneuvers in a pick-and-place operation is considered here. The optimal control formulation incorporates realistic constraints on the joint velocities and accelerations as well as bounds on actuator torque to ensure the solution trajectories are physically realizable. The efficacy of PS optimal control techniques is demonstrated via simulation. The first analysis considers a simple two link, two degree of freedom (2-DOF), three-dimensional motion of an arm with a full development of kinematics, dynamics, constraints, and Pontryagin's necessary conditions [18]. The second analysis uses the same principles to develop trajectories for a single three link, 3-DOF manipulator.

A. PROBLEM FORMULATION FOR A 2-DOF MANIPULATOR

The robotic arms used throughout this study are based on the Cyton Alpha 7D1G from Robai. While not exact, the model assumes homogenous rigid arms and perfect actuators. Dynamics that are more complex can be incorporated into the problem formulation with the same techniques demonstrated below used to formulate the optimal control problem.

1. Modeling Dynamics and Kinematics

The Cyton Alpha is a seven degrees of freedom manipulator with an end effector as pictured in Figure 5 [34]. Specific joints can be locked to simulate fewer degrees of freedom. The first problem formulated is a three dimensional motion of a 2-DOF arm as sketched in Figure 6 where a_i is the offset in the local

x coordinate system and d_i is the offset in the local z direction. θ_1 is the angle of rotation of the base and θ_2 is the angle of the arm with respect to horizontal. Based on the Cyton Alpha specifications, limits were placed on the motor torque for each joint, τ_{imax} and the maximum angular velocity, ω_{imax}. Initial discrepancies between the model dynamics and the actual arm in the laboratory required angular acceleration limits, α_{imax} in order to increase the accuracy of the results. Link 1, the base unit, is modeled as a homogeneous cylinder and the arm is modeled as a thin rod.

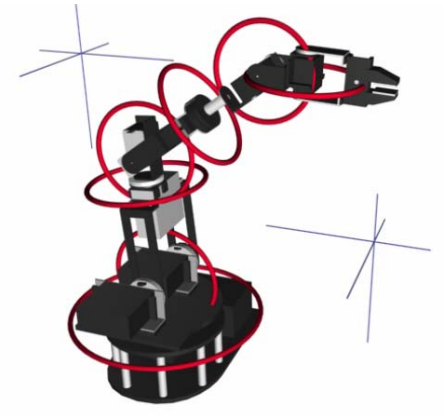

Figure 5. Cyton Alpha 7D1G 7-DOF Robotic Manipulator (From [34])

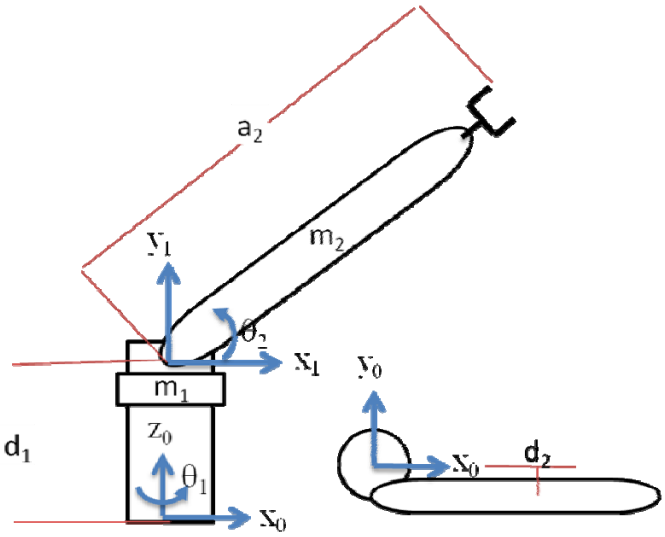

Figure 6. Sketch of 2-DOF Robotic Arm Model

Link 1 (Base)		Link 2 (Arm)	
a_1	0.00 m	a_2	0.48 m
d_1	0.15 m	d_2	0.02 m
m_1	0.150 kg	m_2	0.385 kg
τ_{1max}	2.4 N m	τ_{2max}	2.9 N m
ω_{1max}	7.5 rad/sec	ω_{2max}	7.5 rad/sec
α_{1max}	7.5 rad/sec^2	α_{2max}	10 rad/sec^2

Table 1. 2-DOF Dimensions and Limits

It is usually desirable to map the position of the end effector from joint coordinates to a Cartesian coordinate system. The origin for the coordinate system is the center of the base. A Denavit-Hartenberg homogenous transfer matrix was used to characterize the kinematics of the robotic arm. Using planar rotations and linear displacements Equations (11) and (12) explicitly define the end effector position.

$$\begin{Bmatrix} x_e \\ y_e \\ z_e \end{Bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{pmatrix} a_1 \\ -d_2 \\ d_1 \end{pmatrix} + \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \begin{pmatrix} a_2 \\ 0 \\ 0 \end{pmatrix} \right) \tag{11}$$

$$\begin{aligned} x_e &= a_1\cos\theta_1 + d_2\sin\theta_1 + a_2\cos\theta_1\cos\theta_2 \\ y_e &= a_1\sin\theta_1 - d_2\cos\theta_2 + a_2\sin\theta_1\cos\theta_2 \\ z_e &= d_1 + a_2\sin\theta_2 \end{aligned} \tag{12}$$

While this is the method employed almost universally and will be used throughout this work, other means of deriving the kinematics are available and may prove to be better suited to solving the optimal control problem. Reference [15] has suggested the use of Lie groups and Lie algebras to formulate the kinematics and subsequent dynamics of the system.

The dynamics of the robotic arm were calculated using a Lagrangian formulation of the equations of motion [7,35,36]. The general equation of motion takes the form

$$D(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = \mathbf{Q} \tag{13}$$

where **q** is the vector of generalized coordinates (θ_1 and θ_2 in this case), D is a positive definite inertia matrix, C is the velocity coupling vector, g is the gravitation force vector, and Q is the generalized force vector or the joint torques, τ_1 and τ_2 in this case. The algorithm presented in [35] was used to develop the general equations for this model symbolically using the software program Maple and is presented in the appendix. Using the values from Figure 6, Equation (13) was solved and is shown to three significant figures for the 2-DOF three-dimensional model in Equation (14).

$$\begin{bmatrix} 0.0128 + 0.0126\cos 2\theta_2 & -0.00183\sin 2\theta_2 \\ -0.00183\sin 2\theta_2 & 0.0291 \end{bmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} + \cdots$$
$$\cdots + \begin{bmatrix} -0.0291\dot{\theta}_1\dot{\theta}_2 \sin 2\theta_2 - 0.00183\dot{\theta}_2^2 \cos\theta_2 \\ 0.0146\dot{\theta}_1^2 \sin 2\theta_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.899\cos\theta_2 \end{bmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} \quad (14)$$

2. Formulation of Optimal Control Problem

The dynamics formulated above are the basis for the optimal control problem formulation shown in Equation set (15). Let **x** be the state vector and **u** the control vector. The cost function, *J*, was chosen to minimize the time of the maneuver based on the constraints of the system. Other cost functions can be developed to minimize the control effort (such as a quadratic cost), energy, or some combination of elements. **e** is the endpoint function where the initial angles θ^0 and the final endpoint coordinates [x^f, y^f, z^f] are given by solving Equation (12) for a feasible θ_1 and θ_2. In addition, let **h** be defined as the path constraints on the function that in this case is simply the angle, angular velocity, and torque limits.

$$\mathbf{x} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \in X := \begin{Bmatrix} \theta_1 : -90^\circ \le \theta_1 \le 90^\circ \\ \theta_2 : 45^\circ \le \theta_2 \le 135^\circ \\ \dot{\theta}_1 : -7.5 \le \dot{\theta}_1 \le 7.5 \\ \dot{\theta}_2 : -7.5 \le \dot{\theta}_2 \le 7.5 \end{Bmatrix} \quad \mathbf{u} = \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^T \in U := \begin{Bmatrix} \tau_1 : -2.4 \le \tau_1 \le 2.4 \\ \tau_2 : -2.9 \le \tau_2 \le 2.9 \end{Bmatrix}$$

$$\underset{u}{\text{Minimize}} \qquad J\left[\mathbf{x}(\square), \mathbf{u}(\square), t_f\right] = t_f$$

$$\text{Subject to:} \quad \dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \qquad \text{where} \quad \alpha_i = f(\mathbf{x}, \mathbf{u})$$

$$\mathbf{x}(t_0) = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

$$\mathbf{e}(\mathbf{x}_f, t_f) = \begin{bmatrix} a_1 \cos\theta_1 + d_2 \sin\theta_1 + a_2 \cos\theta_1 \cos\theta_2 - x^f \\ a_1 \sin\theta_1 - d_2 \cos\theta_2 + a_2 \sin\theta_1 \cos\theta_2 - y^f \\ d_1 + a_2 \sin\theta_2 - z^f \\ \omega_1^f \\ \omega_2^f \end{bmatrix} = \begin{bmatrix} \mathbf{0} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{x}^L \\ \mathbf{u}^L \end{bmatrix} \le \mathbf{h}(\mathbf{x}, \mathbf{u}) \le \begin{bmatrix} \mathbf{x}^U \\ \mathbf{u}^U \end{bmatrix}$$

3. Necessary Conditions

To be optimal, the solution must satisfy Pontryagin's necessary conditions. Define the Hamiltonian, *H* as a function of running cost, *F,* the vector of costates, **λ**(t), and the right-hand side of the dynamics, $\dot{\mathbf{x}}(t) = f(x,u,t)$:

$$H(\boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, t) := F + \boldsymbol{\lambda}^T \mathbf{f} \; ; \quad \boldsymbol{\lambda} \in \square^{N_\mathbf{x}}$$

$$H = \lambda_{\theta_1}\omega_1 + \lambda_{\theta_2}\omega_2 + \lambda_{\omega_1}\alpha_1 + \lambda_{\omega_2}\alpha_2 \tag{16}$$

The path constraints must be taken into account as well. For the simplest 2 DOF problem the constraints are the limits of the control and state vectors

defined in Equation set (1515). Equation (17) defines the Lagrangian of the Hamiltonian, $\bar{H}$.

$$\begin{aligned} \bar{H} &:= H + \boldsymbol{\mu}^T \mathbf{h}; \qquad \boldsymbol{\mu} \in \mathbb{R}^{N_h} \\ \bar{H} &= \mu_{\theta_1}\theta_1 + \mu_{\theta_2}\theta_2 + (\lambda_{\theta_1} + \mu_{\omega_1})\omega_1 + (\lambda_{\theta_2} + \mu_{\omega_2})\omega_2 + \lambda_{\omega_1}\alpha_1 + \lambda_{\omega_2}\alpha_2 + \mu_{\tau_1}\tau_1 + \mu_{\tau_2}\tau_2 \end{aligned} \tag{17}$$

Similarly, the Equation (18) defines the Endpoint Lagrangian, $\bar{E}$ where E is the endpoint cost:

$$\begin{aligned} \bar{E}(t_0, t_f, \mathbf{x}_0, \mathbf{x}_f, \mathbf{v}) &:= E + \mathbf{v}^T \mathbf{e}; \quad \mathbf{v} \in \mathbb{R}^{N_e} \\ \bar{E} &= \mathbf{v}_{\mathbf{x}}^T \mathbf{x}_0 + \upsilon_{xf} x_f + \upsilon_{yf} y_f + \upsilon_{zf} z_f + \mathbf{v}_{\boldsymbol{\omega}}^T \boldsymbol{\omega} \end{aligned} \tag{18}$$

Pontryagin's necessary conditions are defined in Equations (19) – (23). The Hamiltonian Minimization Condition (HMC) requires $\frac{\partial \bar{H}}{\partial \mathbf{u}} = 0$.

$$\frac{\partial \bar{H}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial \bar{H}}{\partial \tau_1} \\ \frac{\partial \bar{H}}{\partial \tau_2} \end{pmatrix} = \begin{pmatrix} \lambda_{\omega_1} \frac{\partial \alpha_1}{\partial \tau_1} + \lambda_{\omega_2} \frac{\partial \alpha_2}{\partial \tau_1} + \mu_{\tau_1} \\ \lambda_{\omega_1} \frac{\partial \alpha_1}{\partial \tau_2} + \lambda_{\omega_2} \frac{\partial \alpha_2}{\partial \tau_2} + \mu_{\tau_2} \end{pmatrix}$$

$$\frac{\partial \bar{H}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\lambda_{\omega_1} 14600}{Den} + \frac{\lambda_{\omega_2} 917 \sin\theta_2}{Den} + \mu_{\tau_1} \\ \frac{\lambda_{\omega_1} 917 \sin\theta_2}{Den} + \frac{\lambda_{\omega_2}(7390 + 7280\cos 2\theta_2)}{Den} + \mu_{\tau_2} \end{pmatrix} = [\mathbf{0}] \tag{19}$$

$$Den = 214 + 212\cos 2\theta_2 + 1.68\cos^2\theta_2$$

The adjoint condition requires $-\dot{\boldsymbol{\lambda}} = \frac{\partial \bar{H}}{\partial \mathbf{x}}$.

$$-\dot{\underline{\boldsymbol{\lambda}}} = \begin{pmatrix} \frac{\partial \bar{H}}{\partial \theta_1} \\ \frac{\partial \bar{H}}{\partial \theta_2} \\ \frac{\partial \bar{H}}{\partial \omega_1} \\ \frac{\partial \bar{H}}{\partial \omega_2} \end{pmatrix} = \begin{pmatrix} \lambda_{\omega_1} \frac{\partial \alpha_1}{\partial \theta_1} + \lambda_{\omega_2} \frac{\partial \alpha_2}{\partial \theta_1} + \mu_{\theta_1} \\ \lambda_{\omega_1} \frac{\partial \alpha_1}{\partial \theta_2} + \lambda_{\omega_2} \frac{\partial \alpha_2}{\partial \theta_2} + \mu_{\theta_2} \\ (\lambda_{\theta_1} + \mu_{\omega_1}) + \lambda_{\omega_1} \frac{\partial \alpha_1}{\partial \omega_1} + \lambda_{\omega_2} \frac{\partial \alpha_2}{\partial \omega_1} \\ (\lambda_{\theta_2} + \mu_{\omega_2}) + \lambda_{\omega_1} \frac{\partial \alpha_1}{\partial \omega_2} + \lambda_{\omega_2} \frac{\partial \alpha_2}{\partial \omega_2} \end{pmatrix}$$

$$-\dot{\boldsymbol{\lambda}} = \begin{pmatrix} 0 \\ (\lambda_{\omega_1} + \mu_{\alpha_1}) \frac{\partial \alpha_1}{\partial \theta_2} + (\lambda_{\omega_2} + \mu_{\alpha_2}) \frac{\partial \alpha_2}{\partial \theta_2} + \mu_{\theta_2} \\ (\lambda_{\theta_1} + \mu_{\omega_1}) + \lambda_{\omega_1} \frac{\sin(2\theta_2)(252\omega_2 - 15.9\sin(\theta_2)\omega_1)}{127 + 126\cos(2\theta_2) + \cos^2 \theta_2^2} + \cdots \\ \cdots + \lambda_{\omega_2} \frac{\sin(2\theta_2)(15.9\sin(\theta_2)\omega_2 - 128\omega_1 - 126\omega_1 \cos 2\theta_2)}{127 + 126\cos(2\theta_2) + \cos^2 \theta_2^2} \\ (\lambda_{\theta_2} + \mu_{\omega_2}) + \lambda_{\omega_1} \frac{\sin(2\theta_2)(252\omega_2 - 15.9\sin(\theta_2)\omega_1)}{127 + 126\cos(2\theta_2) + \cos^2 \theta_2^2} + \cdots \\ \cdots + \lambda_{\omega_2} \frac{\sin(\theta_2)(15.9\sin(2\theta_2)\omega_1 + 2\cos(\theta_2)\omega_2)}{127 + 126\cos(2\theta_2) + \cos^2 \theta_2^2} \end{pmatrix} \quad (20)$$

It is valuable to note that $\lambda_{\theta 1}$ is constant and can be used to verify the optimality conditions. The transversality condition requires

$$\boldsymbol{\lambda}(t_f) = \frac{\partial \bar{\bar{E}}}{\partial \mathbf{x}_f} \quad \text{and} \quad \boldsymbol{\lambda}(t_0) = -\frac{\partial \bar{\bar{E}}}{\partial \mathbf{x}_0}$$

$$\boldsymbol{\lambda}(t_0) = -\mathbf{v}_{\mathbf{x}}$$

$$\boldsymbol{\lambda}(t_f) = \begin{pmatrix} \upsilon_{xf} \frac{\partial x_f}{\partial \theta_1} + \upsilon_{xy} \frac{\partial y_f}{\partial \theta_1} + \nu_{zf} \frac{\partial z_f}{\partial \theta_1} \\ \upsilon_{xf} \frac{\partial x_f}{\partial \theta_2} + \upsilon_{xy} \frac{\partial y_f}{\partial \theta_2} + \nu_{zf} \frac{\partial z_f}{\partial \theta_2} \\ \upsilon_{\omega_1} \\ \upsilon_{\omega_2} \end{pmatrix}$$

$$\boldsymbol{\lambda}(t_f) = \begin{pmatrix} \upsilon_{xf}(-0.476\sin\theta_1\cos\theta_2 + 0.02\cos\theta_1) + \cdots \\ \qquad \cdots + \upsilon_{xy}(0.476\cos\theta_1\cos\theta_2 + 0.02\sin\theta_1) \\ \upsilon_{xf}(-0.476\sin\theta_1\sin\theta_2) + \cdots \\ \qquad \cdots + \upsilon_{xy}(-0.476\sin\theta_1\sin\theta_2) + \nu_{zf}(0.476\sin\theta_2) \\ \upsilon_{\omega_1} \\ \upsilon_{\omega_2} \end{pmatrix} \tag{21}$$

The Hamiltonian Value Condition requires:

$$\overline{H}(t_f) = -\frac{\partial \overline{E}}{\partial t_f} = -1 \tag{22}$$

Finally, the Hamiltonian Evolution Equation requires that the optimal control trajectory satisfies Equation (23).

$$\dot{\overline{H}} = \frac{\partial \overline{H}}{\partial t} = 0 \tag{23}$$

The above necessary conditions provide two useful checks for optimality. The Hamiltonian can be analyzed for optimality and should be a constant -1 for all time and $\lambda_{\theta 1}$ is constant.

4. Minimum Time Simulation Set-up and Results

The three-dimensional 2-DOF robotic arm is used to demonstrate the applicability of using pseudospectral techniques to solve the time-optimal trajectories with various endpoint conditions using the software package DIDO. Two trajectories are presented below. To ensure that the endpoint coordinates are reachable within the state bounds, feasible initial and final angles were first selected. The final angles were then mapped to the corresponding final manipulator endpoints using Equation (12). The assumption in this the simplest case is that the final orientation of the manipulator is not a constraint on the system, only that it is at the correct position. It is important to note than even in the 2-DOF system, a given set of joint angles produces a unique point in space, but the reverse is not necessarily true. A point in space may have multiple

solutions in joint space and choosing specific final angles to ensure the feasibility of the maneuver does not guarantee that those angles would be the optimal final state.

To obtain the optimal solution, DIDO was run in normal mode using a bootstrapping technique. The initial run was calculated using a 16 node solution that in turn was used as a guess for a 30 node solution that in turn was used as an initial guess for a 60 node solution. It was found that 60 nodes were sufficient to properly propagate the system using a differential solver such as ODE45 and using the linearly interpolated values of the discrete control trajectory.

Table 2 lists the initial conditions used for two scenarios. The final angles were used to calculate the final endpoint constraints. Scenario 1 consisted of a positive 90° rotation about the base of the arm with the initial and final angle of the second link at 45°. Figure 7 shows the optimal trajectory results of that run. As shown, the time-optimal maneuver was calculated to be a simple rotation about the base. Figure 7 - Figure 9 plot the results for analysis. As shown, *H=-1* and $\lambda_{\theta 1}$ is constant as required by Equations (23) and (20). The minimum time maneuver completes in 0.92 seconds for this problem formulation.

2-DOF Simulation Scenario 1			2-DOF Simulation Scenario 2		
$\theta_1^0 = 0$ $\theta_2^0 = 45^\circ$	$\theta_1^f = 90^\circ$ $\theta_2^f = 45^\circ$	$x_f = 0.020$ $y_f = 0.337$ $z_f = 0.483$	$\theta_1^0 = 0$ $\theta_2^0 = 45^\circ$	$\theta_1^f = -90^\circ$ $\theta_2^f = 45^\circ$	$x_f = -0.020$ $y_f = -0.337$ $z_f = 0.483$

Table 2. 2-DOF Simulation Initial Conditions

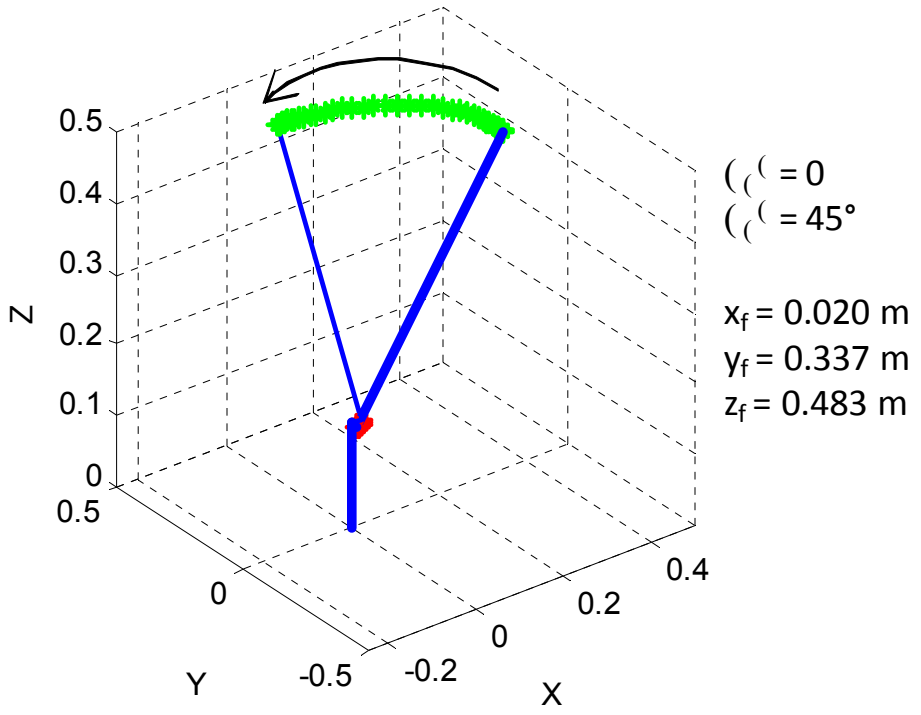

Figure 7. 2 DOF arm simulation, Scenario 1: 90° maneuver.

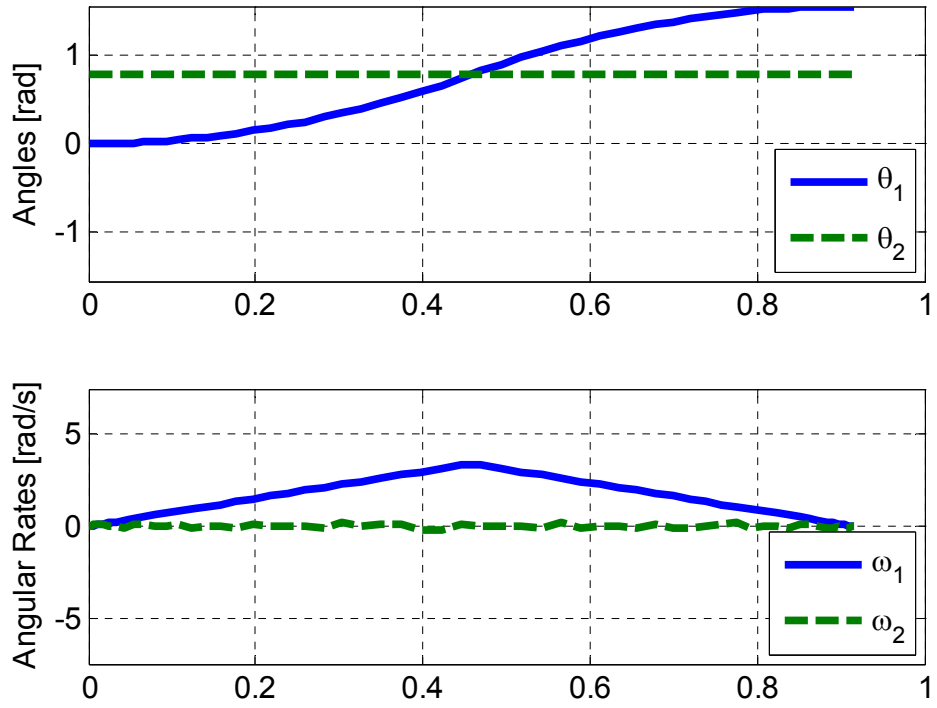

Figure 8. 2-DOF Arm Simulation, Scenario 1 State History

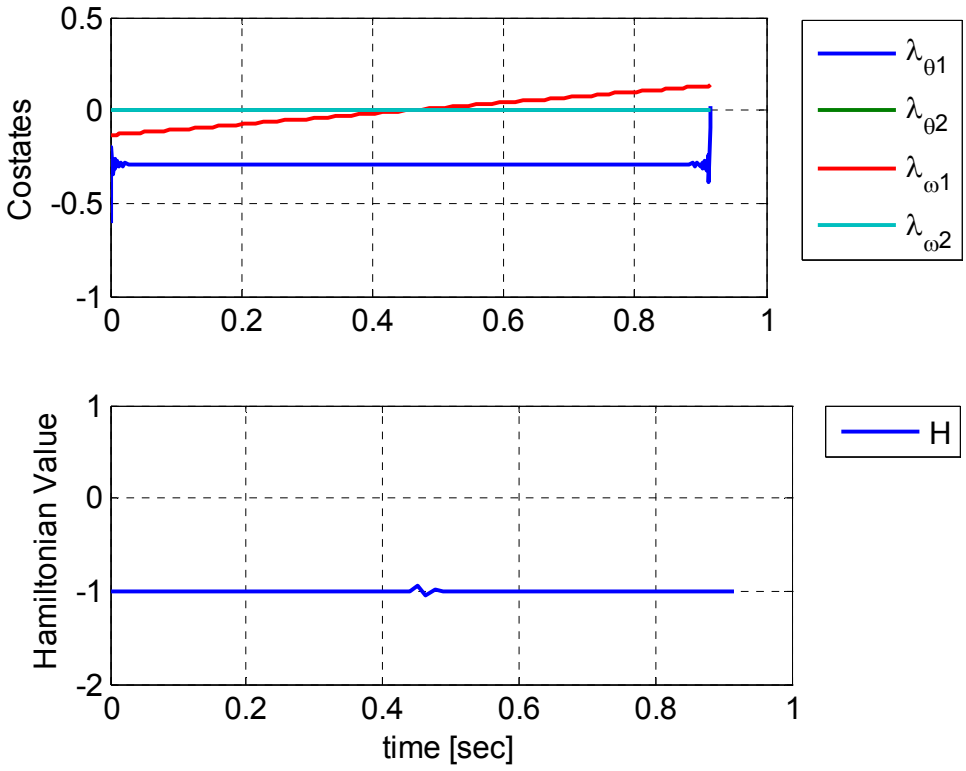

Figure 9. 2-DOF Arm Simulation, Scenario 1: Costates and Hamiltonian

Scenario 2 was designed as a similar maneuver with the arm rotating -90° about the base. Again, the final coordinates of the manipulator end effector were calculated using Equation (11). The results were quite different than the previous scenario as shown in Figure 10 with a maneuver time of 0.880 seconds. Figure 11 shows that the final base angle, θ_1, is not -90° as expected but rather 83.2° (1.45 radians) and the arm elevation, θ_2, is 135°. No assumptions were made as to the trajectory of the arm and it became obvious that the endpoint coordinates could be achieved in two ways and solving the reverse kinematics problem shows this. The pseudospectral solver solved the solution that required the minimum time to accomplish, which in this case was a rotation of 83.2° vice a rotation of 90° due to the offset of the arm from the centerline of the base.

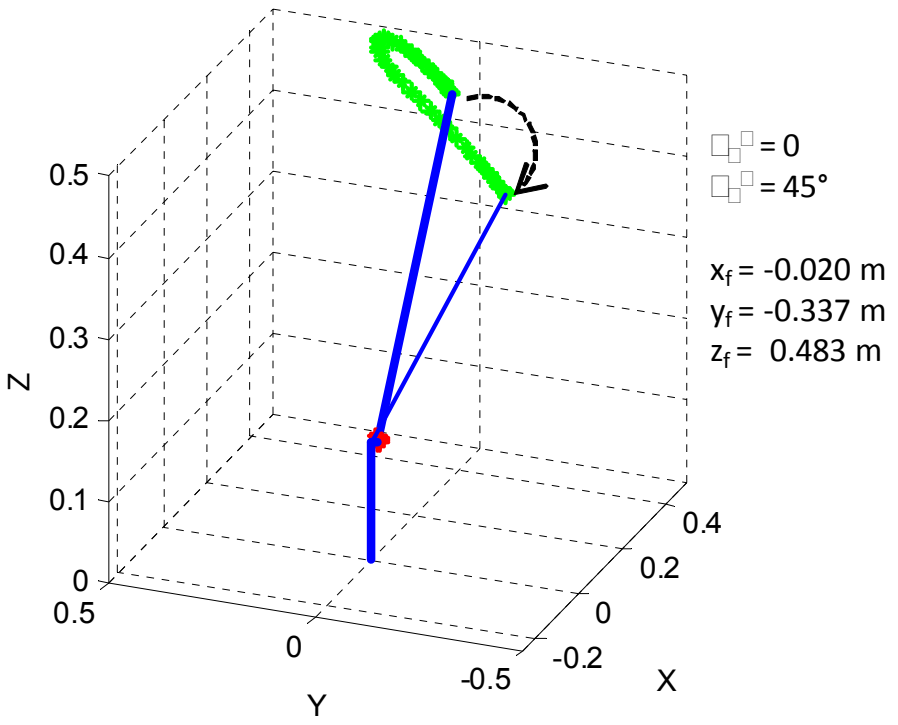

Figure 10. 2-DOF Arm Simulation, Scenario 2: "-90°" maneuver.

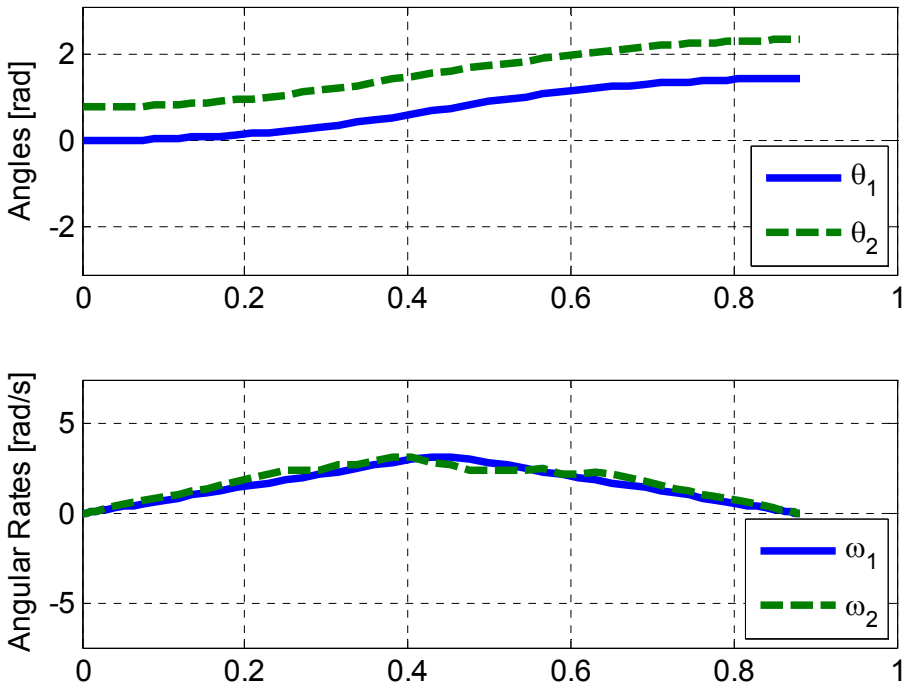

Figure 11. 2-DOF Arm Simulation, Scenario 2: State History

The control torques on both joint actuators were indirectly constrained in the simulation by an added constraint on the angular acceleration based on empirical performance characteristics measured on a physical arm. Nevertheless, the pseudospectral solver produced a control that resembles a bang-bang control for both actuators as shown in Figure 12. However, the torque required to rotate the base was noticeably smaller. The momentum of the arm in combination with the small base torque, was sufficient to rotate the arm the 83.2° required to complete the maneuver.

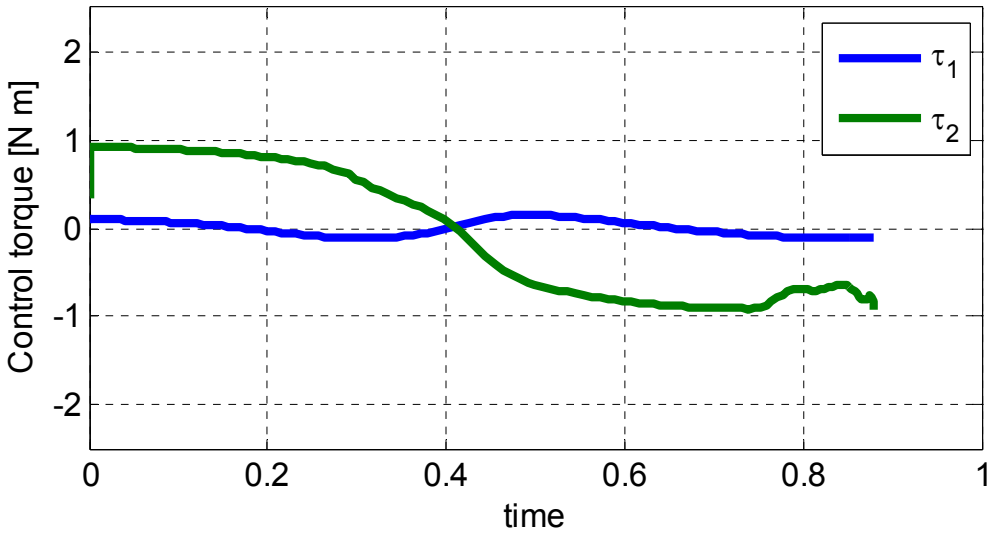

Figure 12. 2-DOF Arm Simulation, Scenario 2: Control Torque

To ensure the feasibility of the trajectory, the control torque was interpolated using a basic linear interpolation function in Matlab and propagated using the same dynamic equations (14) and the ODE45 function at 20 Hz intervals (the minimum sampling rate of the arm). The results are plotted in Figure 13 with the propagated angles and angular velocities overlaid every 0.05 seconds on the DIDO solution.

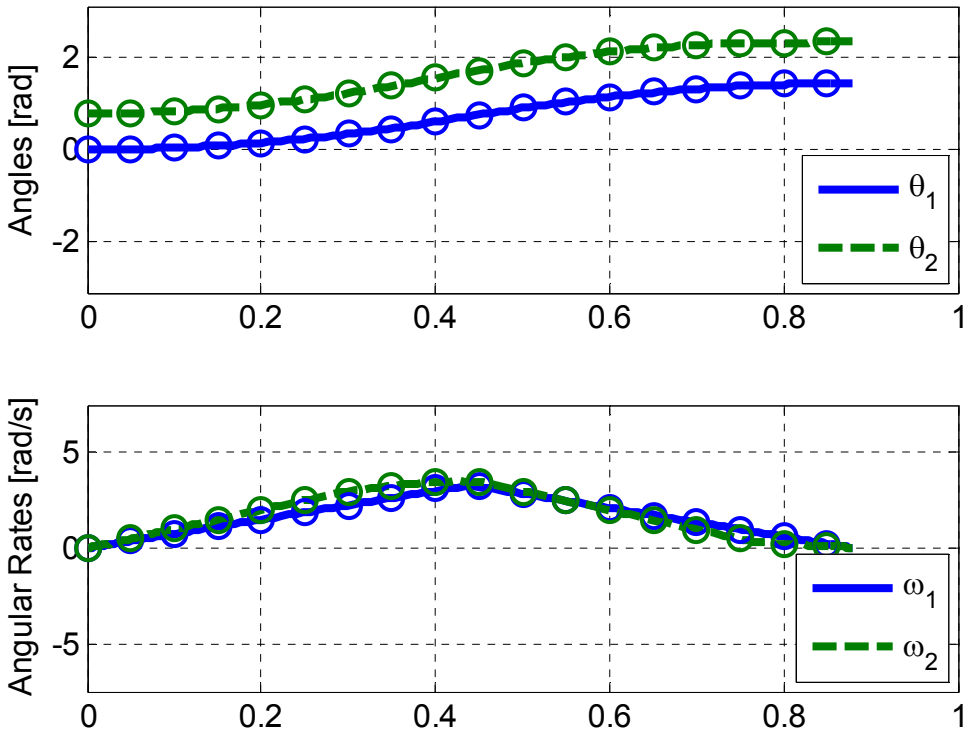

Figure 13. 2 DOF Arm Simulation, Scenario 2: Propagated State Vector.

While relatively simple dynamics were used in the above problem formulation, the method can be used on more complex dynamics that include actuator characteristics and non-rigid links [37]. The PS methods implemented by DIDO would refine the optimal solution as the accuracy of the model increases.

B. PROBLEM FORMULATION FOR 3-DOF MANIPULATOR

1. Modeling Dynamics and Kinematics

The Cyton Alpha arm was also modeled as a 3-DOF manipulator. Figure 14 is a sketch of the configuration used in deriving the equations of motion. The angle of the bottom link was locked at 45°. Maple was used to map the joint positions and end-effector position from link coordinates to Cartesian coordinates similar to the 2-DOF case above. Equation (24) defines the positions of the arm end-effector as functions of θ_i. For the purposes of this simulation, the two free

links have dimensions given in Table 3 and are joined by simple revolute joints with the angular limits defined by Equation set (25). The Equations of motion for this model were developed using the algorithm presented in [35] in form of Equation (13) using Maple.

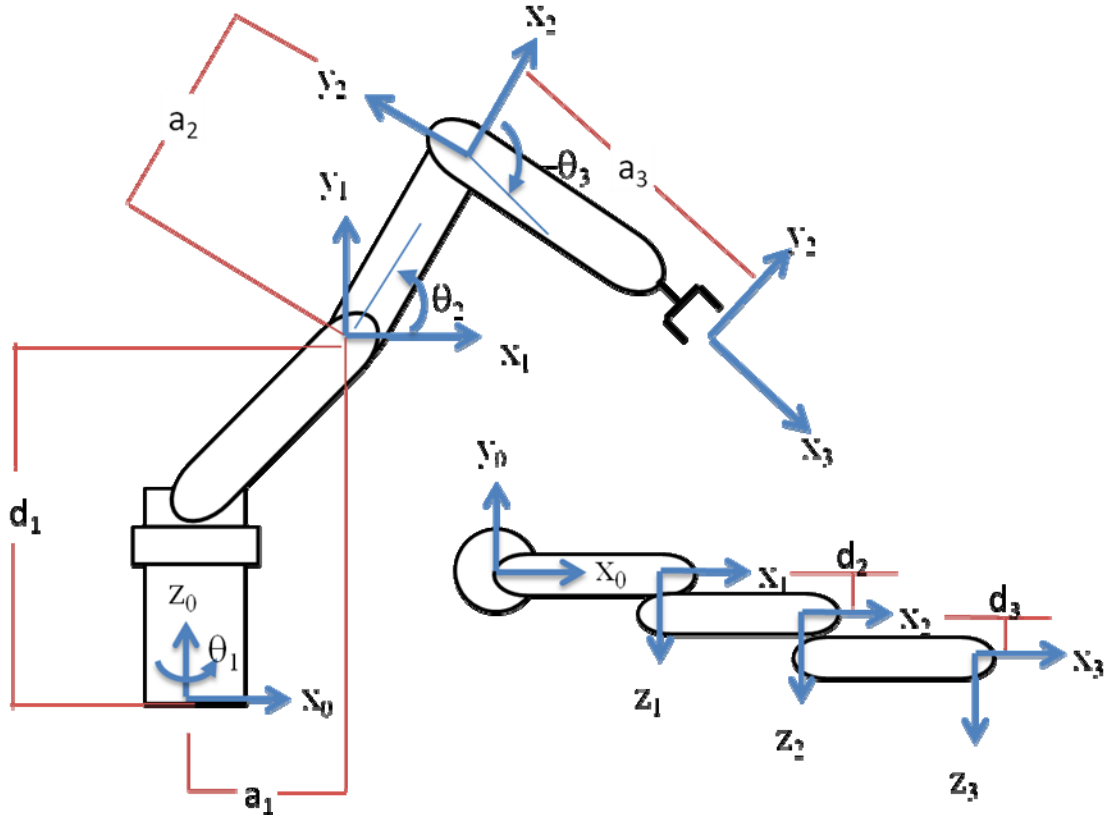

Figure 14. 3 DOF Robotic Arm Configuration

Link 1 (Base+Arm 1)		Link 2		Link 3	
a_1	0.115 m	a_2	0.170 m	a_3	0.140 m
d_1	0.270 m	d_2	0.020 m	d_3	0.010 m
m_1	0.375 kg	m_2	0.100 kg	m_3	0.110 kg

Table 3. 3-DOF Arm Dimensions

$$
\begin{aligned}
x_e &= a_1\cos\theta_1 + (d_2+d_3)\sin\theta_1 + a_2\cos\theta_1\cos\theta_2 + a_3\cos\theta_1\cos(\theta_2+\theta_3)\\
y_e &= a_1\sin\theta_1 - (d_2+d_3)\cos\theta_1 + a_2\sin\theta_1\cos\theta_2 + a_3\sin\theta_1\cos(\theta_2+\theta_3) \qquad (24)\\
z_e &= d_1 + a_2\sin\theta_2 + a_3\sin(\theta_2+\theta_3)
\end{aligned}
$$

2. Formulation of Optimal Control Problem and Necessary Conditions

The optimal control problem builds on the previous 2-DOF example. With the state and control variables, **x** and **u** respectively, defined below, the minimum time problem formulation is shown in Equation set (25). The θ_i and τ_i limits were defined in the Cyton documentation. In order for the links to be kept rigid by intermediate motors, angular acceleration limits, α_i were imposed on the system based on actual tests conducted on the arm. It is important to note that $\ddot{\theta}_i = \alpha_i$ in the inertial reference frame. While not ideal, the above described formulation is to demonstrate the feasibility of the pseudospectral methods and an exact model was deemed beyond the scope of this investigation.

$$\mathbf{x} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \in X := \begin{Bmatrix} \theta_1 : -90^\circ \le \theta_1 \le 90^\circ \\ \theta_2 : -40^\circ \le \theta_2 \le 130^\circ \\ \theta_3 : -85^\circ \le \theta_3 \le 85^\circ \\ \dot{\theta}_1 : -7.5 \le \dot{\theta}_1 \le 7.5 \\ \dot{\theta}_2 : -7.5 \le \dot{\theta}_2 \le 7.5 \\ \dot{\theta}_3 : -5.8 \le \dot{\theta}_3 \le 5.8 \end{Bmatrix} \quad \mathbf{u} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \in U := \begin{Bmatrix} \tau_1 : -2.4 \le \tau_1 \le 2.4 \\ \tau_2 : -2.9 \le \tau_2 \le 2.9 \\ \tau_3 : -0.5 \le \tau_3 \le 0.5 \end{Bmatrix}$$

$$\begin{aligned} &\underset{u}{\text{Min}} \qquad J\left[\mathbf{x}(\square), \mathbf{u}(\square), t_f\right] = t_f \\ &\text{Subject to:} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \quad \text{where } \alpha_i = f(\mathbf{x}, \mathbf{u}) \qquad (25) \\ &\mathbf{x}(t_0) = \begin{bmatrix} \theta_1^0 & \theta_2^0 & \theta_3^0 & 0 & 0 & 0 \end{bmatrix}^T \end{aligned}$$

$$\mathbf{e}(\mathbf{x}_f, t_f) = \begin{bmatrix} x_e - x^f \\ y_e - y^f \\ z_e - z^f \\ \omega_1^f \\ \omega_2^f \\ \omega_3^f \end{bmatrix} = [\mathbf{0}]$$

$$\begin{bmatrix} \mathbf{x}^L \\ \mathbf{u}^L \\ \boldsymbol{\alpha}^L \end{bmatrix} \le \mathbf{h}(\mathbf{x}, \mathbf{u}) \le \begin{bmatrix} \mathbf{x}^U \\ \mathbf{u}^U \\ \boldsymbol{\alpha}^U \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} -10 \le \alpha_1 \le 10 \\ -7.5 \le \alpha_2 \le 7.5 \\ -7.5 \le \alpha_3 \le 7.5 \end{bmatrix}$$

Just as in the 2-DOF problem, Pontryagin's necessary conditions for optimality were derived for the 3-DOF arm. These conditions are similar in form to Equations (16) - (23). The only two useful results of the necessary condition analysis are that the H=-1 for all time, t and $\lambda_{\theta 1}$ is constant.

3. Simulation Results

A number of simulations using this problem formulation were conducted using DIDO. For each simulation, the initial angles and the final manipulator coordinates were defined. The final position of the robotic end-effector was chosen such that the final joint angles were at their upper limits. Figure 15 shows the results. Just as they were calculated in the 2-DOF model, the manipulator coordinates were calculated using Equation (24) given a set of feasible joint angles. The results are similar to those from the 2-DOF problem. Figure 16 and Figure 17 show the computed state and control trajectories for this problem.

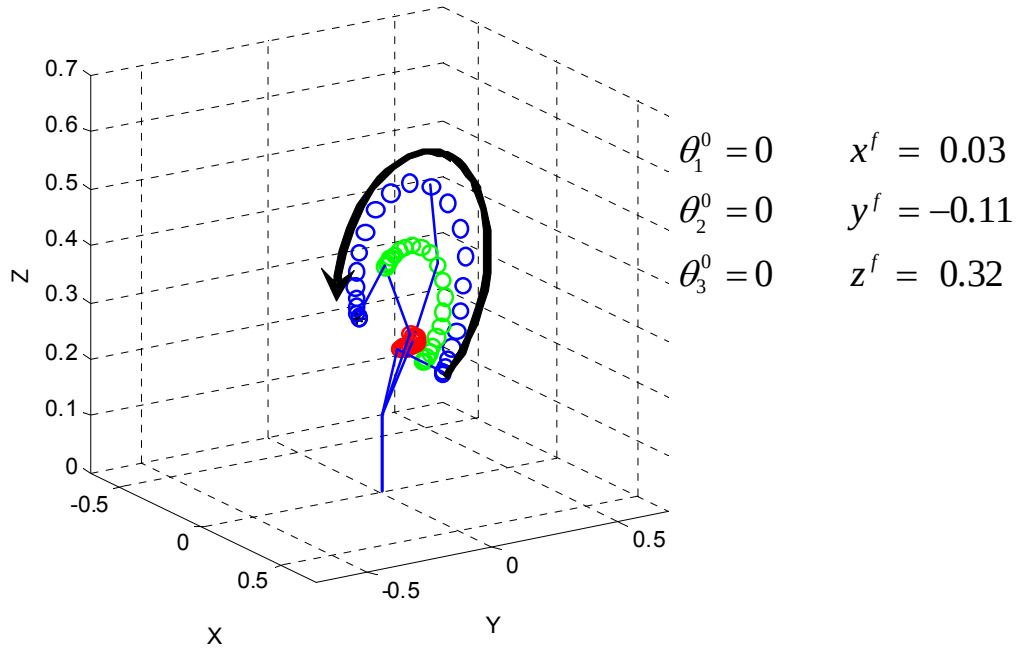

Figure 15. Single Arm 3 Degree of Freedom (3-DOF) Optimal Time Maneuver

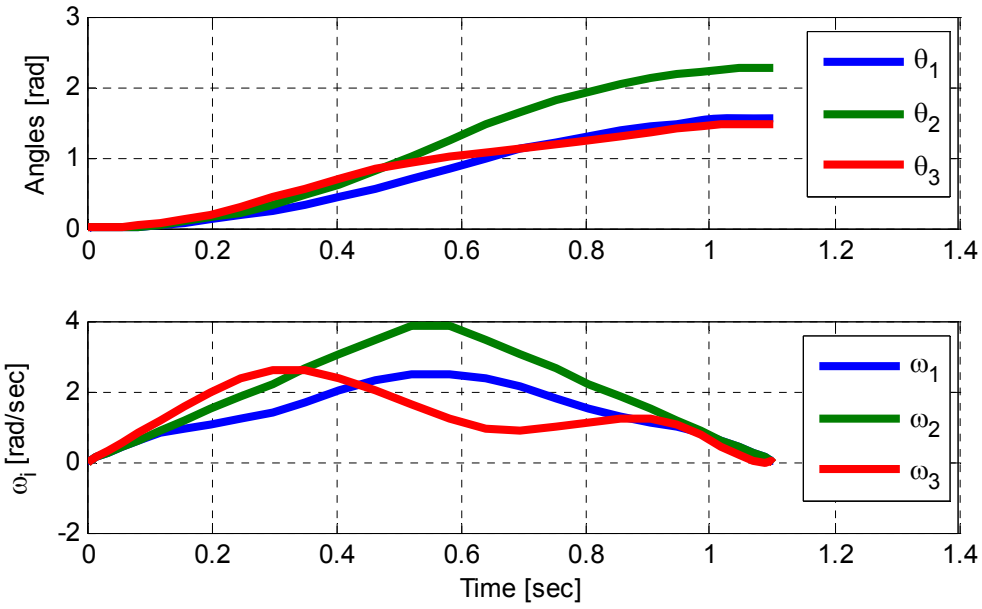

Figure 16. Single Arm 3-DOF Simulation State Trajectory

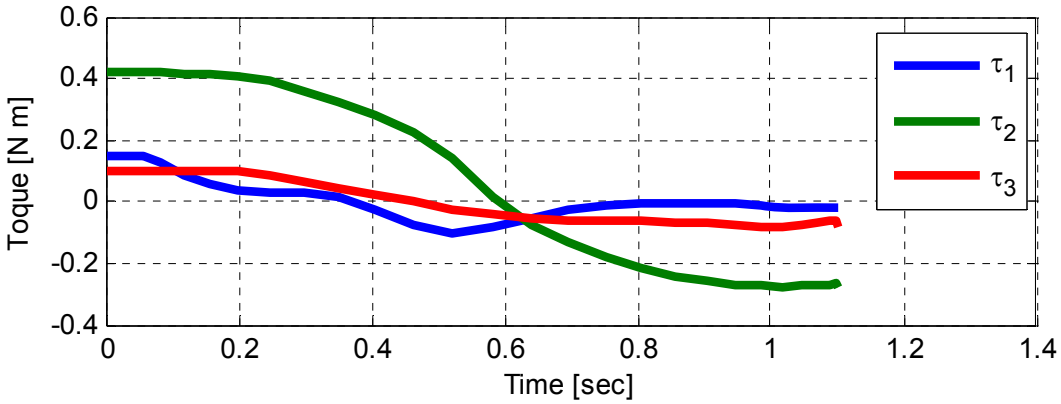

Figure 17. Single Arm 3-DOF Simulation Control Trajectory

The increase in complexity from the 2-DOF to 3-DOF problem is evident from the time duration of the pseudospectral calculations. The runtime for the 2-DOF problem averaged a few seconds, while the 3-DOF problem took about one minute. More importantly was that DIDO required an initial guess for some of the endpoint conditions in order to find a feasible solution. For this problem, a two point guess was used with the initial and final angles and a final time of 2 seconds. The angular velocity and control torque were assumed to be 0 for the guess. DIDO was able to solve the optimal control problem and find a feasible solution. Future versions of DIDO will incorporate more advanced guess-free algorithms such as those discussed in [38] that may alleviate this requirement.

C. DUAL ARM TRAJECTORY PLANNING

The ultimate goal of this investigation is the simultaneous path planning of multiple robotic manipulators. As such, the next step taken was to formulate the minimum time problem using two arms and analyze the results. Below is the 3-DOF problem formulation with simulation results from both 2-DOF and 3-DOF arms. No attempt was made to avoid arm interference, but only to demonstrate the ability for pseudospectral methods to calculate dual arm trajectories and compare to single arm trajectories.

1. Problem Formulation

The dynamics and endpoints of the multi-arm problem are formulated similarly to the single arm problem. The dual arm formulation is presented below for the 3-DOF robotic motion case. In this case, the arms are identical copies of each other, but this does not need to be the case. Equation Set (26) defines the endpoint coordinates for each arm *a* and *b*. Arm *b* is offset from arm *a* and the origin by the vector $[x_0^b, y_0^b, z_0^b]^T$.

$$
\begin{aligned}
x_e^a &= a_1\cos\theta_1^a + (d_2+d_3)\sin\theta_1^a + a_2\cos\theta_1^a\cos\theta_2^a + a_3\cos\theta_1^a\cos(\theta_2^a+\theta_3^a)\\
y_e^a &= a_1\sin\theta_1^a - (d_2+d_3)\cos\theta_1^a + a_2\sin\theta_1^a\cos\theta_2^a + a_3\sin\theta_1^a\cos(\theta_2^a+\theta_3^a)\\
z_e^a &= d_1 + a_2\sin\theta_2^a + a_3\sin(\theta_2^a+\theta_3^a)\\
x_e^b &= x_0^b + a_1\cos\theta_1^b + (d_2+d_3)\sin\theta_1^b + a_2\cos\theta_1^b\cos\theta_2^b + a_3\cos\theta_1^b\cos(\theta_2^b+\theta_3^b)\\
y_e^b &= y_0^b + a_1\sin\theta_1^b - (d_2+d_3)\cos\theta_1^b + a_2\sin\theta_1^b\cos\theta_2^b + a_3\sin\theta_1^b\cos(\theta_2^b+\theta_3^b)\\
z_e^b &= d_1 + a_2\sin\theta_2^b + a_3\sin(\theta_2^b+\theta_3^b)
\end{aligned}
\tag{26}
$$

Equation sets (27) and (28) show that the dynamics of the problem are completely uncoupled in its formulation.

$$
\mathbf{x} = \begin{bmatrix}
\theta_1^a\\ \theta_2^a\\ \theta_3^a\\ \theta_1^b\\ \theta_2^b\\ \theta_3^b\\ \omega_1^a\\ \omega_2^a\\ \omega_3^a\\ \omega_1^b\\ \omega_2^b\\ \omega_3^b
\end{bmatrix} \in X := \begin{Bmatrix}
\theta_1 : -90^\circ \le \theta_1 \le 90^\circ\\
\theta_2 : -40^\circ \le \theta_2 \le 130^\circ\\
\theta_3 : -85^\circ \le \theta_3 \le 85^\circ\\
\omega_1 : -7.5 \le \omega_1 \le 7.5\\
\omega_2 : -7.5 \le \omega_2 \le 7.5\\
\omega_3 : -5.8 \le \omega_3 \le 5.8
\end{Bmatrix} \quad
\mathbf{u} = \begin{bmatrix}
\tau_1^a\\ \tau_2^a\\ \tau_3^a\\ \tau_1^b\\ \tau_2^b\\ \tau_3^b
\end{bmatrix} \in U := \begin{Bmatrix}
\tau_1 : -2.4 \le \tau_1 \le 2.4\\
\tau_2 : -2.9 \le \tau_2 \le 2.9\\
\tau_3 : -0.5 \le \tau_3 \le 0.5
\end{Bmatrix} \tag{27}
$$

The optimal control problem is now defined as follows:

$$\underset{u}{\text{Minimize}} \quad J\left[\mathbf{x}(\square),\mathbf{u}(\square),t_f\right]=t_f$$

$$\text{Subject to:} \quad \dot{\mathbf{x}}(t)=\begin{bmatrix}\dot{\boldsymbol{\theta}}^a\\ \dot{\boldsymbol{\theta}}^b\\ \dot{\boldsymbol{\omega}}^a\\ \dot{\boldsymbol{\omega}}^b\end{bmatrix}=\begin{bmatrix}\boldsymbol{\omega}^a\\ \boldsymbol{\omega}^b\\ \boldsymbol{\alpha}^a\\ \boldsymbol{\alpha}^b\end{bmatrix} \quad \text{where } \boldsymbol{\alpha}^i=f(\boldsymbol{\theta}^i,\boldsymbol{\omega}^i,\boldsymbol{\tau}^i)$$

$$\mathbf{x}(t_0)=\begin{bmatrix}\theta_1^{a0} & \theta_2^{a0} & \theta_3^{a0} & \theta_1^{b0} & \theta_2^{b0} & \theta_3^{b0} & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T$$

$$\mathbf{e}(\mathbf{x}_f,t_f)=\begin{bmatrix}x_e^a-x^{af}\\ y_e^a-y^{af}\\ z_e^a-z^{af}\\ x_e^b-x^{bf}\\ y_e^b-y^{af}\\ z_e^b-z^{bf}\\ \omega_1^{af}\\ \omega_2^{af}\\ \omega_3^{af}\\ \omega_1^{bf}\\ \omega_2^{bf}\\ \omega_3^{bf}\end{bmatrix}=\begin{bmatrix}\mathbf{0}\end{bmatrix} \qquad \begin{bmatrix}\mathbf{x}^L\\ \mathbf{u}^L\\ \boldsymbol{\alpha}^L\end{bmatrix}\le\mathbf{h}(\mathbf{x},\mathbf{u})\le\begin{bmatrix}\mathbf{x}^U\\ \mathbf{u}^U\\ \boldsymbol{\alpha}^U\end{bmatrix}$$

$$\text{where} \quad \begin{bmatrix}-10\le\alpha_1^i\le 10\\ -7.5\le\alpha_2^i\le 7.5\\ -7.5\le\alpha_3^i\le 7.5\end{bmatrix} \tag{28}$$

2. Simulation Results

Adding the second arm increases the complexity of the problem and for some endpoint conditions, a simple two-point guess was required for DIDO to converge to a solution. A bootstrapping technique was used with a 16 node solution being calculated then used as a guess for a 30 node and then a 60 node solution. The second arm was offset from the origin by 30 cm in the –y direction. Figure 18 plots the simulation results for the two arms with the given endpoint conditions. Figure 19 plots the Hamiltonian for the system and verifies the necessary Hamiltonian Value Condition.

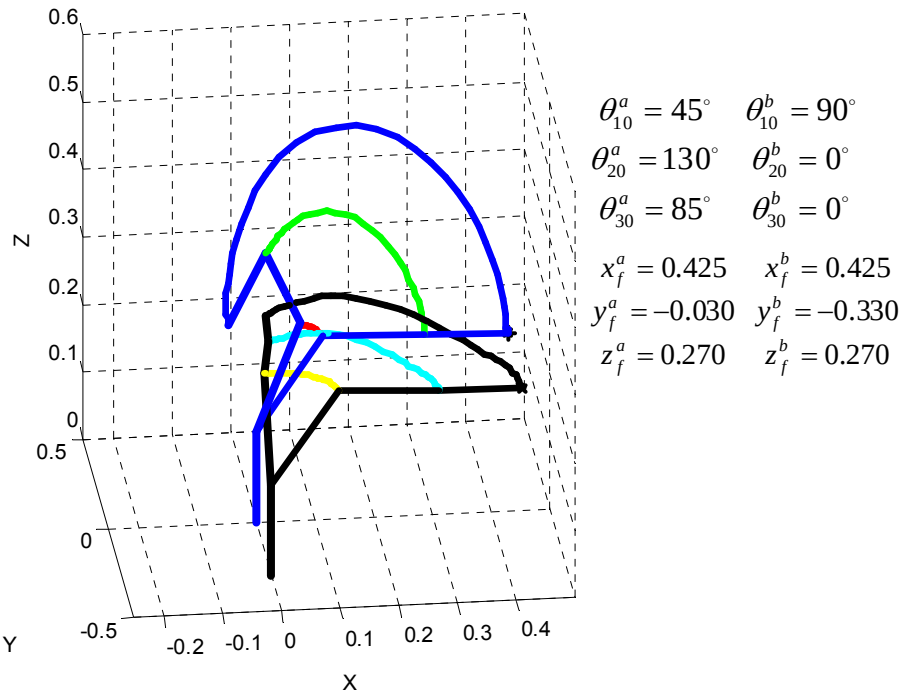

Figure 18. Dual Arm 3-DOF Simulation (No Obstacle Avoidance)

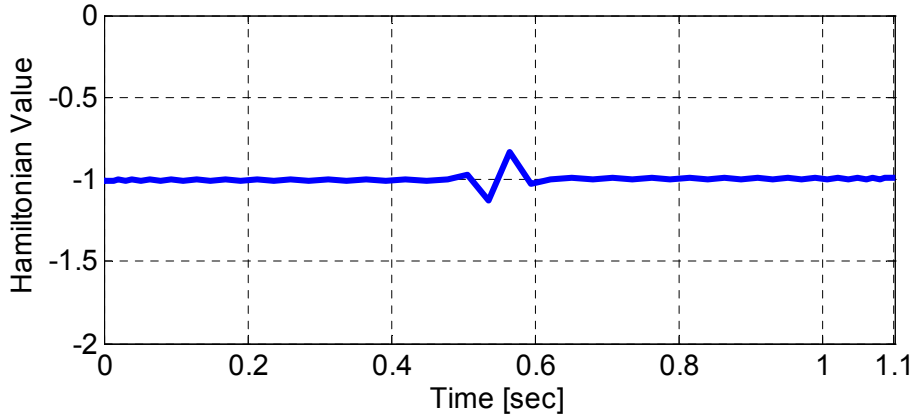

Figure 19. Dual Arm 3-DOF (No Obstacle Avoidance) Hamiltonian

These results were derived using minimum time as the cost function. While all the joint angles are coupled, generally there is a joint that limits the minimum time, in this case θ_2 for arm A and θ_1 for arm B. In addition, one arm will limit the minimum time of the entire multi-arm system. It is important to note that the minimum time for the system to complete the maneuver is dictated by

the arm whose time-optimal maneuver takes longer to complete. The trajectory of the other arms and joints will technically meet the minimum time condition regardless of the path it takes so long as it arrives at its endpoint conditions at t_f.

Figure 20 plots the differences that can occur between the dual arm and single arm minimum time maneuvers. The solid lines in the figures are the single arm solutions and the dashed lines are from the dual arm system. The minimum time defining joint θ_2 for arm A is nearly identical for both the single and dual arm algorithms. The other joints, while not following exactly the same trajectory, did find the same minimum time route to the endpoint. Arm B, which took 0.92 seconds to complete its maneuver in the single arm computation follows a similar path in the dual arm algorithm, but at a slower rate.

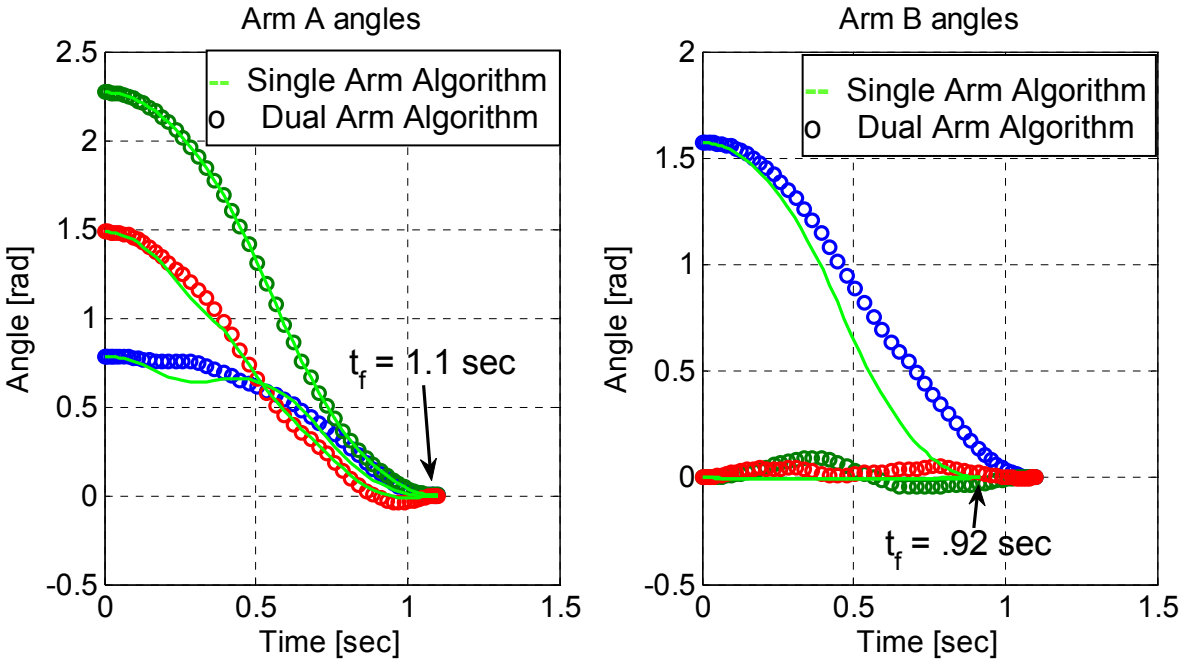

Figure 20. Comparison of Dual Arm Algorithm and Single Arm Algorithm

While technically the results are time-optimal, there is no measure of efficiency in the computation of the angles of arms that are not limited by the system minimum time. A good example is the oscillations in arm B's θ_1 and θ_2 in the dual arm algorithm. One remedy is to solve for the optimal trajectory in two

steps. First, solve the minimum time problem, then use that minimum time to bound the horizon of the problem and solve a minimum energy or quadratic problem among the various possible minimum time trajectories. Another solution is to add some measurement of efficiency such as a minimum energy or minimum control to the cost function to find a minimum time solution that also maximizes the efficiency of the system. The modified cost function could take the form of Equation (29) where [A] is a diagonal matrix of weighting factors. Assuming t_f is sufficiently large compared to the running cost, the minimum time solution can be found while simultaneously maximizing the efficiency of the solution.

$$J = t_f + \int_0^{t_f} \mathbf{u}^T \left[\mathrm{A}\right]\mathbf{u} \tag{29}$$

The results of rerunning the same endpoint conditions using Equation (29) as the cost function with unity weighting are presented in Figure 21. A number of conclusions can be made from this plot. First, it illustrates that there are multiple minimum time paths, particularly for those joints that do not limit the minimum time. The final time using the new cost function is the same to within 10^{-3} second while decreasing quadratic control cost by 26.3%. This improvement factor was found by numerically integrating the calculated control vector of the two scenarios. In addition, these results demonstrate the effect of gravity on the system. In Figure 20, arm B's second and third links remain nearly parallel as θ_2 and θ_3 remain at nearly 0 throughout the maneuver. However, including the quadratic cost demonstrates that the minimum control torque required to complete the maneuver is not the parallel motion. θ_2 and θ_3 in Figure 21 actually drop below parallel to minimize the total control torque over the course of the maneuver.

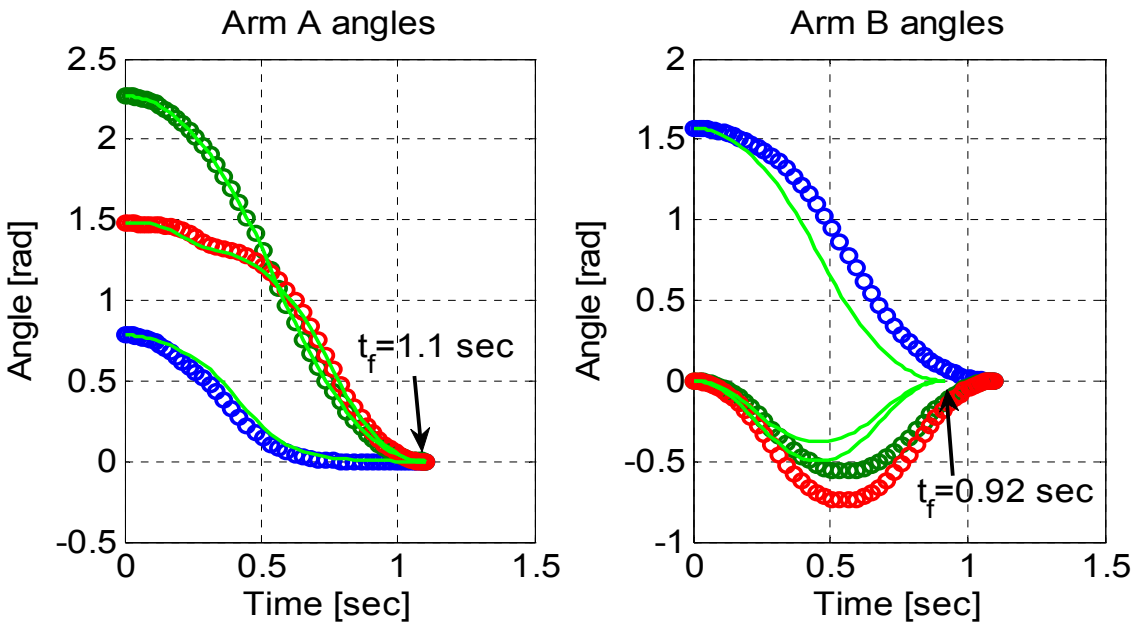

Figure 21. Comparison of Dual Arm Algorithm and Single Arm Algorithms Including Minimum Effort

Changing the cost function, *J,* to Equation (29), does alter some of the necessary conditions, but does not change the two useful conditions used in this analysis to check for optimality: $H(t)$=-1 and $\lambda_{\theta 1}$=constant. What it also does not do is simultaneously find minimum time solution for the arm that does not limit the minimum time for the system. A viable solution for this using pseudospectral methods is not presented here but poses an interesting path for future research.

D. AN ALTERNATE PROBLEM FORMULATION

While the standard problem formulation used above finds an optimal solution, there are other ways of formulating this optimization problem. One such alternate method that was investigated was to reformulate the problem statement by eliminating all trigonometric functions. Rather than defining the state variables in terms of θ_i and ω_i, let the state variables be defined by the sine and cosine of the angles. While this increases the dimension of the state variables, there are

no trigonometric functions in the dynamics which may be computationally advantageous . The state **x,** control **u** and dynamics $\dot{\mathbf{x}}$ for each arm are defined for the 2-DOF problem as

$$\mathbf{x} = \begin{pmatrix} S_{\theta_1} \\ C_{\theta_1} \\ S_{\theta_2} \\ C_{\theta_2} \\ \omega_1 \\ \omega_2 \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} \qquad \dot{\mathbf{x}} = \begin{pmatrix} \dot{\theta}_1 C_{\theta_1} \\ -\dot{\theta}_1 S_{\theta_1} \\ \dot{\theta}_2 C_{\theta_2} \\ -\dot{\theta}_2 S_{\theta_2} \\ \dot{\omega}_1 \\ \dot{\omega}_2 \end{pmatrix} = \begin{pmatrix} \omega_1 C_{\theta_1} \\ -\omega_1 S_{\theta_1} \\ \omega_2 C_{\theta_2} \\ -\omega_2 S_{\theta_2} \\ \alpha_1(\mathbf{x},\mathbf{u}) \\ \alpha_2(\mathbf{x},\mathbf{u}) \end{pmatrix}$$

Because the states are no longer angles, the following additional path constraints are required on the system:

$$C_{\theta_1}^2 + S_{\theta_1}^2 - 1 = 0$$
$$C_{\theta_2}^2 + S_{\theta_2}^2 - 1 = 0$$

The bounds on the states based on the physical limitations of the system must also be reformulated:

$$0 \le C_{\theta_1} \le 1 \qquad -\frac{\sqrt{2}}{2} \le C_{\theta 2} \le \frac{\sqrt{2}}{2}$$
$$\frac{\sqrt{2}}{2} \le S_{\theta 1} \le 1 \qquad -1 \le S_{\theta 2} \le 1$$

The endpoints must also be redefined by the new state variables:

$$\mathbf{e}_i = \begin{pmatrix} \sin\theta_1^i \\ \cos\theta_1^i \\ \sin\theta_2^i \\ \cos\theta_2^i \end{pmatrix} \qquad \mathbf{e}_f = \begin{pmatrix} x_f\left(C_{\theta_1}^f, S_{\theta_1}^f, C_{\theta_2}^f, S_{\theta_2}^f\right) \\ y_f\left(C_{\theta_1}^f, S_{\theta_1}^f, C_{\theta_2}^f, S_{\theta_2}^f\right) \\ z_f\left(C_{\theta_1}^f, S_{\theta_1}^f, C_{\theta_2}^f, S_{\theta_2}^f\right) \end{pmatrix} \qquad \boldsymbol{\omega}_i = \boldsymbol{\omega}_f = 0$$

Using the same endpoint conditions in section 2-A, the alternate problem formulation was solved using Equation (29) as the cost function in DIDO.

Regardless of cost function, t_f = 0.92 seconds which corresponds to the minimum time solution of the +90° rotation from scenario 1 above. Figure 22 illustrates the calculated system trajectory.

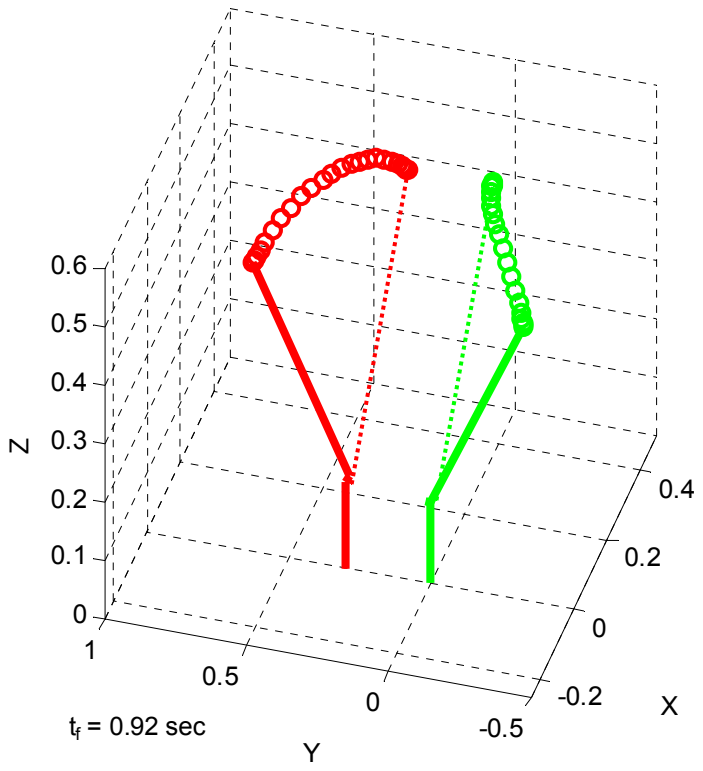

Figure 22. 2 Arm, 2-DOF Simulation Using Alternate Problem Formulation

This result shows that other proper problem formulations will result in the same trajectory as the original problem formulation. Figure 23 compares the calculated joint angle trajectories for the two problem formulations. It is apparent that regardless of the problem formulation, DIDO computed a very similar solution. However, there are preferred formulations for numerical computation efficiency. Figure 24 demonstrares that the original problem formulation, for this particular model, is better scaled and can be solved an order of magnitude faster.

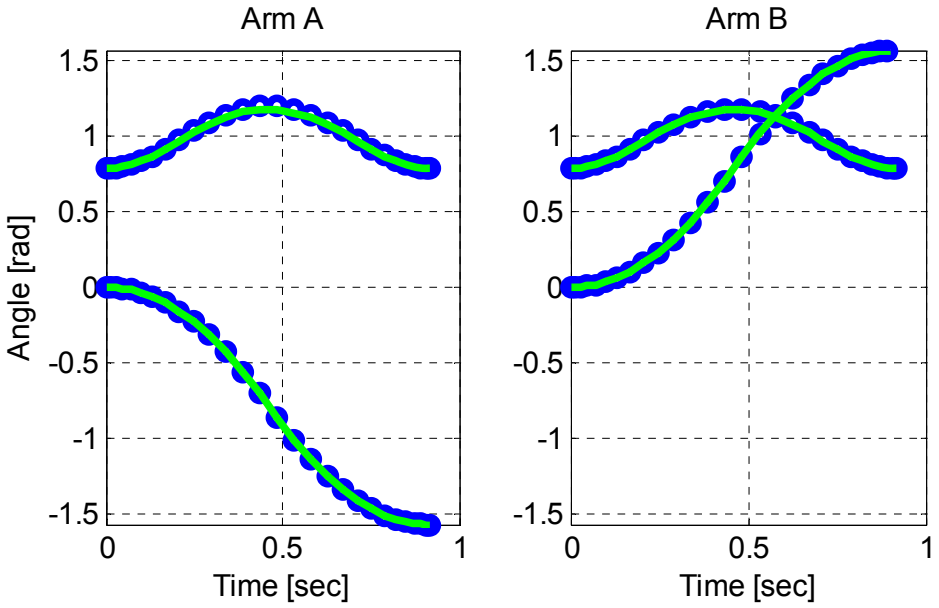

Figure 23. Comparison of Original and Alternate Problem Formulation

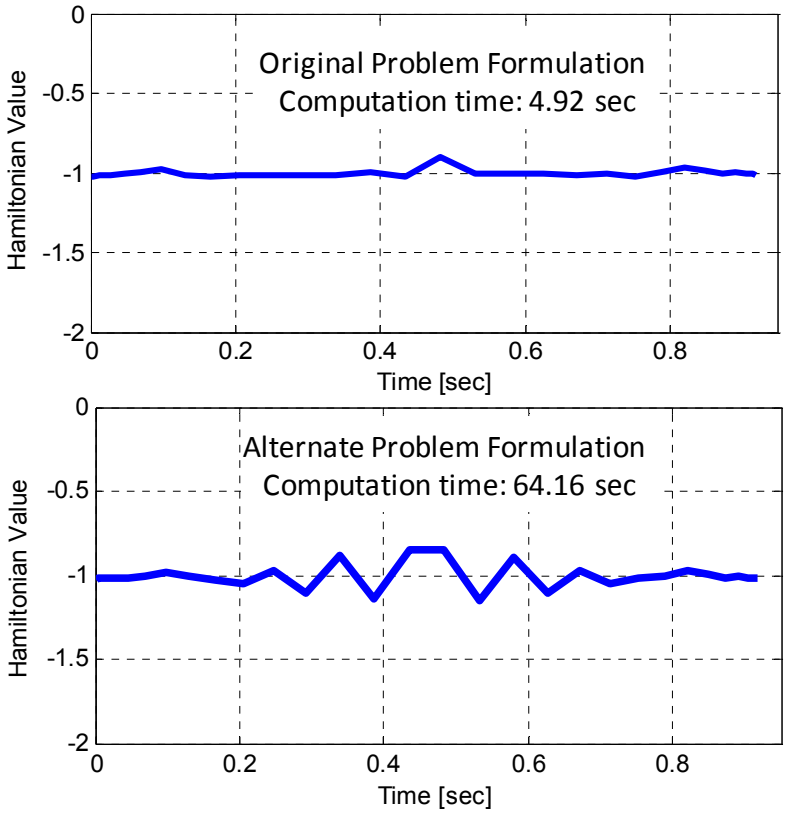

Figure 24. 30 Node Hamiltonian Values for Different Problem Formulations

III. OBSTACLE AVOIDANCE

A. BACKGROUND

Of major importance in robotic trajectory planning is solving the obstacle avoidance problem. In the most simplistic terms, the issue is one of determining the distance between each arm and any potential obstacle. A number of studies have been published on obstacle avoidance in motion planning of robotic vehicles [1, 12, 28 ,39, 40] where the obstacle is mathematically modeled as an enclosed polygon with the vehicle as a point in space and time. A path constraint that restricts the point from entering the polygon by a sufficient buffer ensures collision avoidance is included in the algorithm. While this is useful in a number of autonomous motion planning problems, it does not account for a set of continuous points such as arms of finite length. A different path constraint formulation is required.

In the most general sense, each manipulator link and obstacle can be modeled as some rigid geometric shape and the minimum distance between each link and each potential collision must be calculated at each point in time. Solving the obstacle avoidance problem for a continuous set of points rather than a discrete point in space and time complicates the problem. In the simplest case, each link of each arm can be modeled as a line segment. The minimum distance between every two links that have the potential to collide must be computed at each time and incorporated into the path constraint. The optimal control problem increases in complexity, but pseudospectral methods can still be used to solve the trajectory with this path constraint and a given minimum distance between links.

1. Minimum Distance between Two Continuous Lines

Before discussing the obstacle avoidance algorithm, it is necessary to first determine the minimum distance between two static arms. Each link is modeled

as a line segment using the parametric equations of a line in $\mathbb{R}^3$. Define $\mathbf{p}^1$ and $\mathbf{p}^2$ as the two endpoints of an individual line segment:

$$\mathbf{p}^1 = \begin{Bmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{Bmatrix} \qquad \mathbf{p}^2 = \begin{Bmatrix} p_x^2 \\ p_y^2 \\ p_z^2 \end{Bmatrix}$$

These values are then used to define the following values for lines a and b:

$$\begin{aligned} \mathbf{a}_0 &= \mathbf{p}_a^1 & \mathbf{a} &= \mathbf{p}_a^2 - \mathbf{p}_a^1 \\ \mathbf{b}_0 &= \mathbf{p}_b^1 \quad \text{and} & \mathbf{b} &= \mathbf{p}_b^2 - \mathbf{p}_b^1 \end{aligned}$$

The equations for two links modeled as a line segment where t and s are parametric variables take the form:

$$\begin{aligned} \bar{\mathbf{a}} &= \mathbf{a}_0 + \mathbf{a}t \qquad & t &\in [0,1] \\ \bar{\mathbf{b}} &= \mathbf{b}_0 + \mathbf{b}s \qquad & s &\in [0,1] \end{aligned} \tag{30}$$

For any two values for t and s, the distance between the vectors $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ is the norm of the vector difference between them.

$$d = \left\| \bar{\mathbf{a}} - \bar{\mathbf{b}} \right\|_2 = \left[\left[\left(\mathbf{a}_0 + \mathbf{a}t \right) - \left(\mathbf{b}_0 + \mathbf{b}s \right) \right]^T \left[\left(\mathbf{a}_0 + \mathbf{a}t \right) - \left(\mathbf{b}_0 + \mathbf{b}s \right) \right] \right]^{1/2} \tag{31}$$

It is more convenient to calculate the square of the distance. Squaring Equation (31) and grouping terms yields:

$$\begin{aligned} d^2 = \mathbf{a}^T\mathbf{a}t^2 + 2\left(\mathbf{a}^T\mathbf{a}_0 - \mathbf{a}^T\mathbf{b}_0\right)t - 2\mathbf{a}^T\mathbf{b}st + \cdots \\ \cdots + 2\left(\mathbf{b}^T\mathbf{b}_0 - \mathbf{b}^T\mathbf{a}_0\right)s + \mathbf{b}^T\mathbf{b}s^2 + \mathbf{a}_0^T\mathbf{a}_0 - 2\mathbf{a}_0^T\mathbf{b}_0 + \mathbf{b}_0^T\mathbf{b}_0 \end{aligned} \tag{32}$$

The coefficients of Equation (32) are scalar values. Let

$$
\begin{aligned}
A &= \mathbf{a}^T\mathbf{a} \\
B &= 2\left(\mathbf{a}^T\mathbf{a}_0 - \mathbf{a}^T\mathbf{b}_0\right) \\
C &= 2\mathbf{a}^T\mathbf{b} \\
D &= 2\left(\mathbf{b}^T\mathbf{b}_0 - \mathbf{b}^T\mathbf{a}_0\right) \\
E &= \mathbf{b}^T\mathbf{b} \\
F &= \mathbf{a}_0^T\mathbf{a}_0 - 2\mathbf{a}_0^T\mathbf{b}_0 + \mathbf{b}_0^T\mathbf{b}_0
\end{aligned}
\tag{33}
$$

Substituting Equation set (33) into (32) gives

$$d^2 = At^2 + Bt - Cst + Ds + Es^2 + F \tag{34}$$

Equation (34) is a parabaloid in $\mathbb{R}^3$ with A > 0 and E > 0. The minimum distance for two infinite lines is found by calculating the minimum distance over all points $t,s \in \mathbb{R}$. This is an unconstrained optimization problem with the cost function Y = d^2 and the optimization variable **x** = [t, s]^T or

$$\underset{\mathbf{x}\in\mathbb{R}^2}{\text{Minimize}} \quad Y(t,s) = At^2 + Bt - Cst + Ds + Es^2 + F \tag{35}$$

The minimum must satisfy both the stationary condition and convexity condition:

$$\left.\frac{\partial Y}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} = 0 \qquad \left.\frac{\partial^2 Y}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathbf{x}^*} \geq 0$$

where **x*** are the parametric coordinates of the minimum distance between the two lines. The stationary condition for Equation (35) is

$$\frac{\partial Y}{\partial \mathbf{x}} = \begin{Bmatrix} \dfrac{\partial Y}{\partial t} \\ \dfrac{\partial Y}{\partial s} \end{Bmatrix} = \begin{Bmatrix} 2At + B - Cs \\ -Ct + D + 2Es \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{36}$$

and solving for **x*** yields

$$s^* = \frac{2AD + BC}{(C^2 - 4AE)} \qquad t^* = \frac{2BE + CD}{(C^2 - 4AE)} \tag{37}$$

Likewise, the convexity condition takes the form

$$\frac{\partial^2 Y}{\partial \mathbf{x}^2} = \begin{bmatrix} \dfrac{\partial^2 Y}{\partial t^2} & \dfrac{\partial^2 Y}{\partial t \partial s} \\ \dfrac{\partial^2 Y}{\partial s \partial t} & \dfrac{\partial^2 Y}{\partial s^2} \end{bmatrix} = \begin{bmatrix} 2A & -C \\ -C & 2E \end{bmatrix} \tag{38}$$

Equation (38) is semi-positive definite and meets the convexity condition when 4AE- $C^2 \geq 0$. This is true for all t and s and can be seen by breaking it into the vector components:

$$\begin{aligned} 4AE - C^2 &= 4(\mathbf{a}^T\mathbf{a})(\mathbf{b}^T\mathbf{b}) - 4(\mathbf{a}^T\mathbf{b})(\mathbf{a}^T\mathbf{b}) \\ &= 4\left[|\mathbf{a}|^2 |\mathbf{b}|^2 - \left(|\mathbf{a}||\mathbf{b}|\cos\theta\right)^2 \right] \geq 0 \end{aligned}$$

It is interesting to note that a unique solution only exists when 4AE- $C^2 \neq 0$. When 4AE-C^2=0, the two arms are parallel and there is an infinite set of solutions. Substituting Equation (37) into (35) gives the minimum distance between two infinite lines.

2. Minimum Distance between Two Line Segments

While the discussion on the minimum distance between infinite lines is useful to this problem, the minimum distance between two arm segments is a box-constrained optimization problem:

$$\begin{aligned} &\min Y(t,s) = At^2 + Bt - Cst + Ds + Es^2 + F \\ &\text{subj to} \quad 0 \leq \mathrm{t} \leq 1 \\ &\qquad\quad\; 0 \leq \mathrm{s} \leq 1 \end{aligned} \tag{39}$$

The Lagrangian of Equation (39) is now defined as

$$\bar{Y}(\mathbf{x}, \boldsymbol{\lambda}) = At^2 + Bt - Cst + Ds + Es^2 + F + \lambda_t \mathrm{t} + \lambda_s \mathrm{s}$$

and the first order necessary condition is

$$\left.\frac{\partial \bar{Y}}{\partial \mathbf{x}}\right|_{\min} = \begin{Bmatrix} \frac{\partial \bar{Y}}{\partial t} \\ \frac{\partial \bar{Y}}{\partial s} \end{Bmatrix} = \begin{Bmatrix} 2At + B - Cs + \lambda_t \\ -Ct + D + 2Es + \lambda_s \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{40}$$

This is a more complicated problem to solve. The parametric coordinates for the minimum distance are now functions of the Lagrange multipliers λ_t and λ_s.

$$t\big|_{\min d^2} = \frac{2EB + CD + 2E\lambda_t + C\lambda_s}{\left(C^2 - 4AE\right)} \qquad s\big|_{\min d^2} = \frac{2AD + CB + C\lambda_t + 2A\lambda_s}{\left(C^2 - 4AE\right)} \tag{41}$$

The Lagrange multipliers in Equation (40) and (41) must be found and satisfy the Karush-Kuhn-Tucker (KKT) conditions:

$$\lambda_t \begin{cases} \le 0 \\ = 0 \\ \ge 0 \end{cases} \text{ for } \begin{matrix} t = 0 \\ 0 \le t \le 1 \\ t = 1 \end{matrix} \text{ and } \lambda_s \begin{cases} \le 0 \\ = 0 \\ \ge 0 \end{cases} \text{ for } \begin{matrix} s = 0 \\ 0 \le s \le 1 \\ s = 1 \end{matrix} \tag{42}$$

λ_i cannot be solved analytically in a general sense and other methods must be used to evaluate the endpoints of the segment.

References [41] and [42] develop geometric algorithms to solve the minimum distance between line segments which are summarized below. The first step is to calculate 4AE- C^2 to find out if the line segments are parallel. Next, the minimum parameters t* and s* for two infinite lines that are defined by extending the two line segments are calculated. If those values are both between 0 and 1, then those are the coordinates of the minimum distance and can be used to calculate *d* Equation (34). If t* is outside the range [0,1] and s* is within the range, then the t_{min} is located at the endpoint closest to t*. Using that endpoint condition as a constant, Equation (36) can then be solved for s_{min}. If s_{min} is now outside the range [0,1] then s is set to the endpoint closest to s_{min} and a new t_{min} is computed. The reverse is also true. This approach can be applied if the line segments are parallel and if both global minimums are outside of the range.

Figure 25 demonstrates the algorithm that can be applied to determine parametric coordinates that define the minimum distance between two line segments using values from Equations (33) and (37). While this algorithm efficiently calculates the minimum distance between two static lines, it requires nine if-then statements and is not necessarily a continuously differentiable function over time.

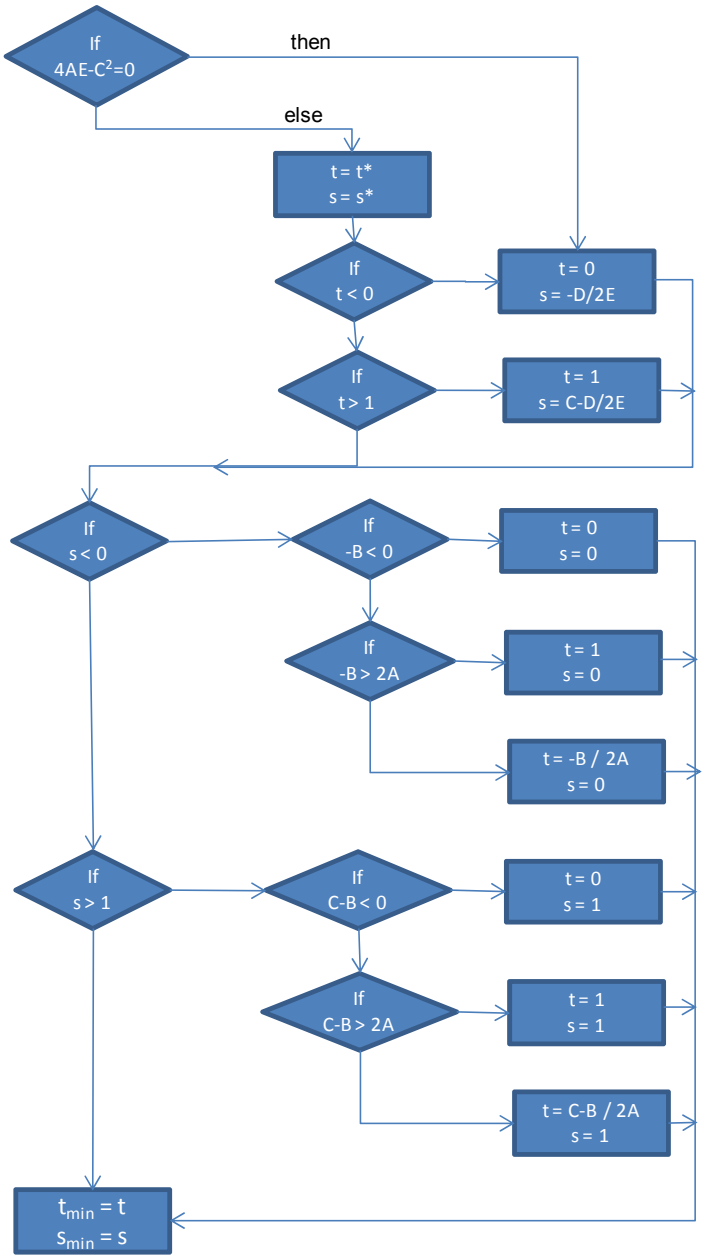

Figure 25. Geometric Algorithm for Min. Distance Between Line Segments.

B. STATIC OBSTACLE AVOIDANCE

1. 2-DOF Case

The above algorithm was used as a path constraint to define the minimum distance between the centerline of an arm and the centerline of a cylinder defined by the respective endpoints. DIDO allows the path constraint to be a function that incorporates the complexity of the geometric analysis. The problem formulation is identical to Equation set (1515) except the added path constraint $d_{\min}^2 \le d^2$ is included where d is the instantaneous distance between the arm and the obstacle and d_{min} is the pre-defined obstacle avoidance distance. For the purposes of simulation, $d_{\min}$= 5 cm. The centerline of the obstacle was placed 45 cm above the base of the arm and oriented along the y axis of the Cartesian coordinate system. Once again, an initial 16 node solution was calculated and then used as a guess for a more refined 60 node solution in order to interpolate a feasible control trajectory when propagating the solution using ODE45 in Matlab. The relevant initial and final conditions of the simulation are included in Table 4.

θ_{10} = - 60°	$[x_{ef}, y_{ef}, z_{ef}]$ = [.186, .281, .483]	$[x_1, y_1, z_1]_{obs}$ = [-0.5, 0, .55]
θ_{20} = 45°	$d_{\min}$ = 5 cm	$[x_2, y_2, z_2]_{obs}$ =[0.5, 0, .55]

Table 4. Example Endpoint Conditions for Single 2-DOF Arm with Static Obstacle Avoidance

Figure 26a demonstrates the solution for the defined maneuver without imposing the obstacle avoidance path constraint and using the cost function

$$J = t_f + .5\int_0^{t_f} \mathbf{u}^T\mathbf{u}$$

As the figure illustrates, the arm passes through the centerline of the obstacle. Figure 26b demonstrates the same maneuver with the collision avoidance algorithm active. The trajectory of the arm changes to allow a minimum 5 cm buffer between the arm and line. Figure 27 -Figure 29 provide the

corresponding state and control trajectories, the costate values over time, the Hamiltonian value, and the variation of the minimum distance between the arm and the obstacle during the obstacle avoidance maneuver.

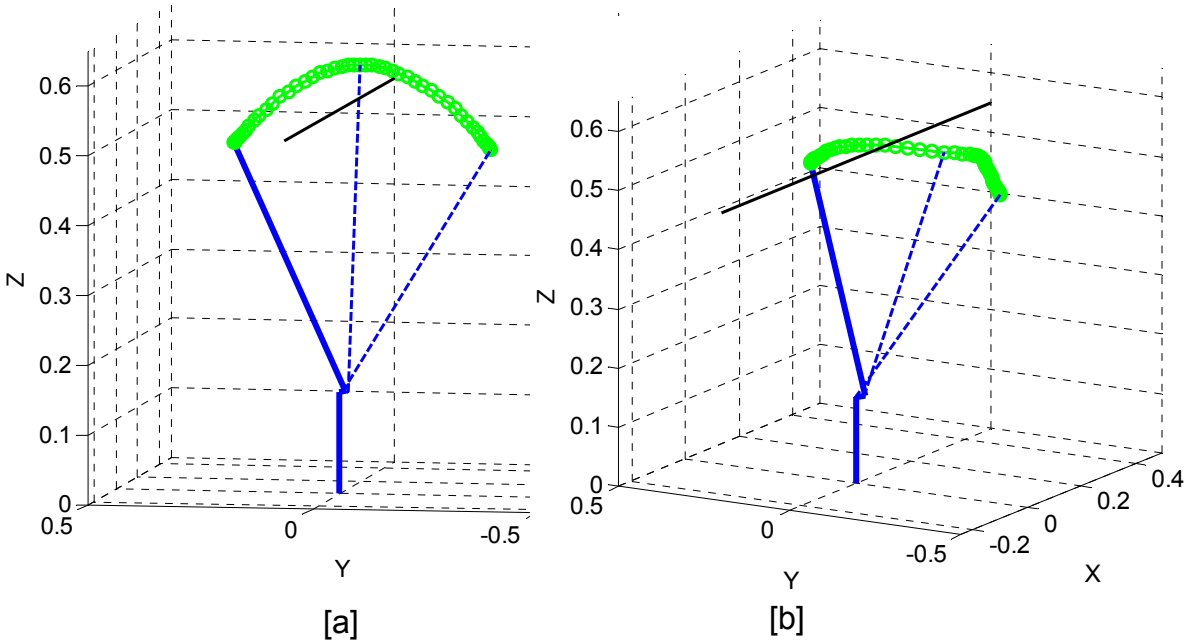

Figure 26. 2-DOF Maneuver Without and With Static Obstacle Avoidance

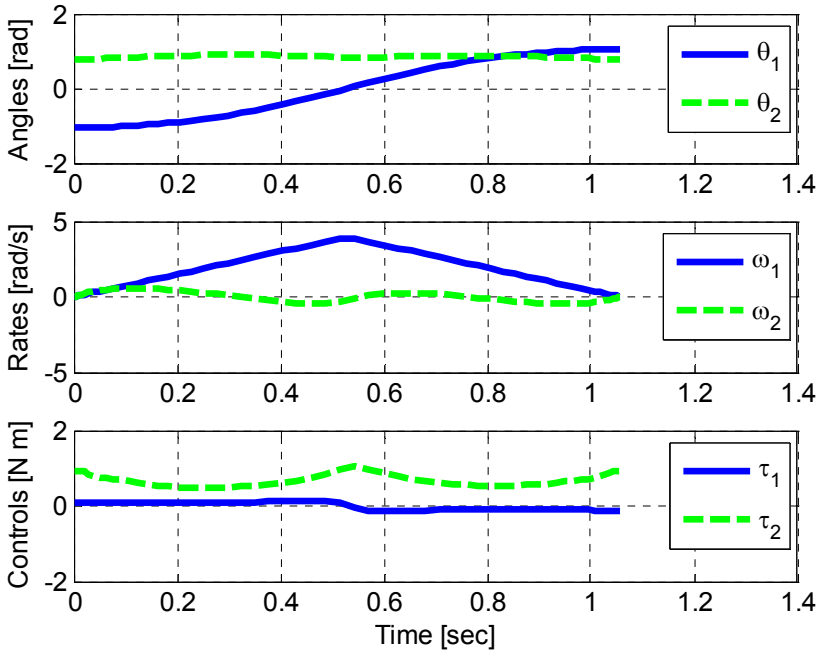

Figure 27. 2-DOF Motion with Static Obstacle Avoidance: State and Control Trajectories

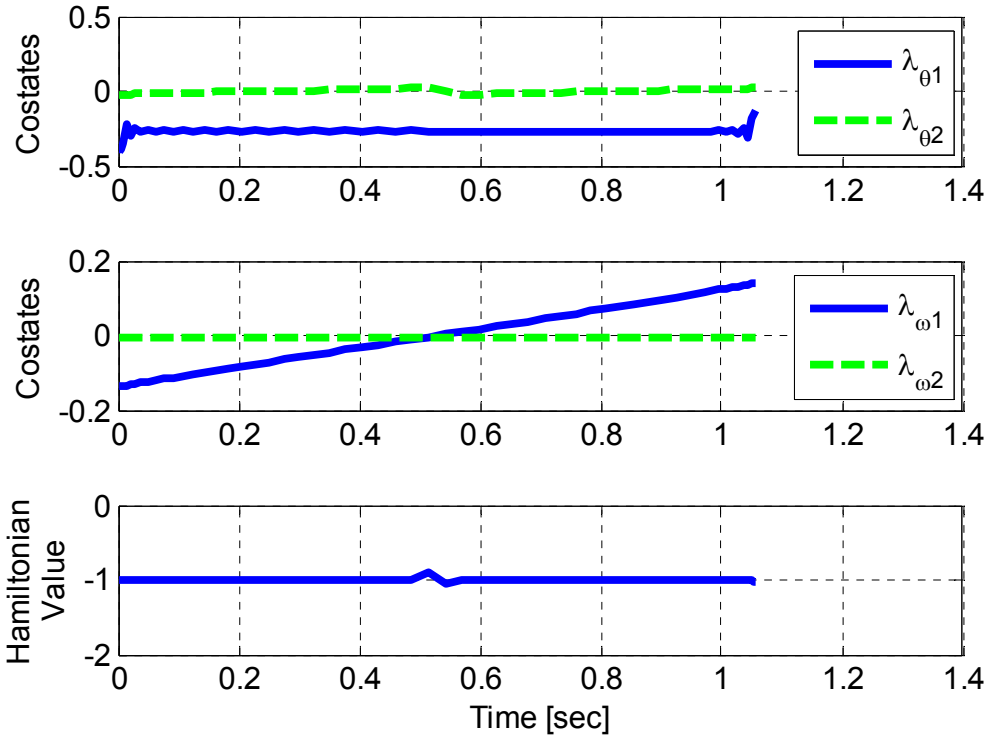

Figure 28. 2-DOF Motion with Static Obstacle Avoidance: Costate and Hamiltonian Trajectories

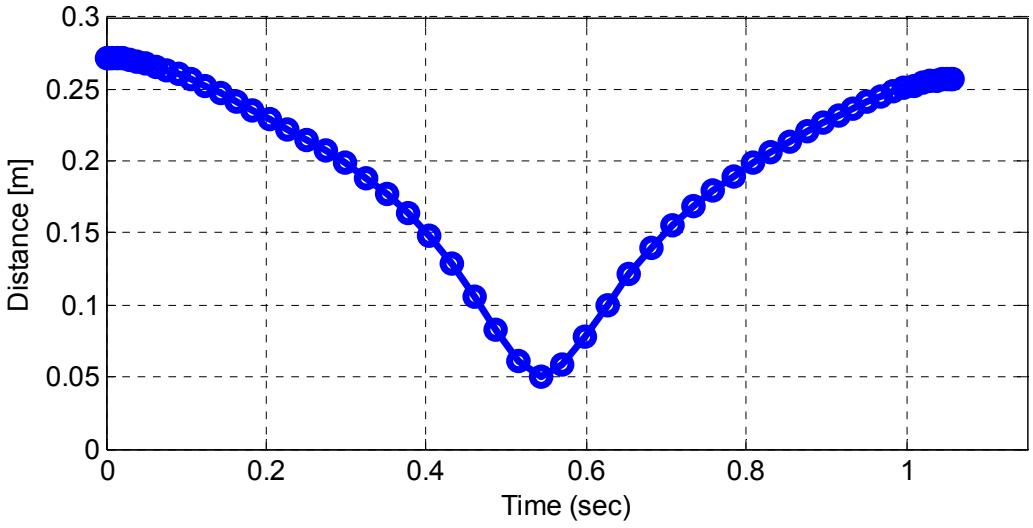

Figure 29. 2-DOF Motion with Static Obstacle Avoidance: Distance between Link and Obstacle (5 cm Buffer)

2. 3-DOF Case

Similar to the 2-DOF case, the 3-DOF problem formulation is identical to Equation Set (25) with the added complexity of the path constraint imposed to guarantee a minimum distance between both links and the obstacle. The path constraint for this problem includes

$$d_{1\min}^2 \le d_1^2$$
$$d_{2\min}^2 \le d_2^2$$

where d_{i_min} are the minimum distances between the obstacle and each respective link. The joint positions must be computed at each node in time using the kinematic model and these points define the line segments used to compute the minimum distance.

θ_{10} = 80°	$[x_{ef}, y_{ef}, z_{ef}]$ = [.029, -.334, .049]	$[x_1, y_1, z_1]_{obs}$ = [-1, 0, 0.4]
θ_{20} = 45°	d_{1min} = 0.05 m	$[x_2, y_2, z_2]_{obs}$ = [1, 0, 0.4]
θ_{20} = 0°	d_{2min} = 0.05 m	

Table 5. Example Endpoint Conditions for Single 3-DOF Arm with Static Obstacle Avoidance

Figure 30a demonstrates the minimum time solution for the above maneuver without the obstacle avoidance algorithm. As the figure illustrates, the arm passes through the centerline of the line used to model the obstacle. Figure 30b demonstrates the same maneuver with the collision avoidance algorithm active. The trajectory of the arm changes to allow a minimum 5 cm buffer between the arm and static cylinder. The maneuver time for both simulations was 1.22 seconds. The difference in the trajectory can be attributed to the weighted quadratic cost function. Figure 31 - Figure 33 are plots of the solution trajectories. Of particular interest are the Hamiltonian and $\lambda_{\theta 1}$ that still satisfy the necessary condition derived in the original 3-DOF case.

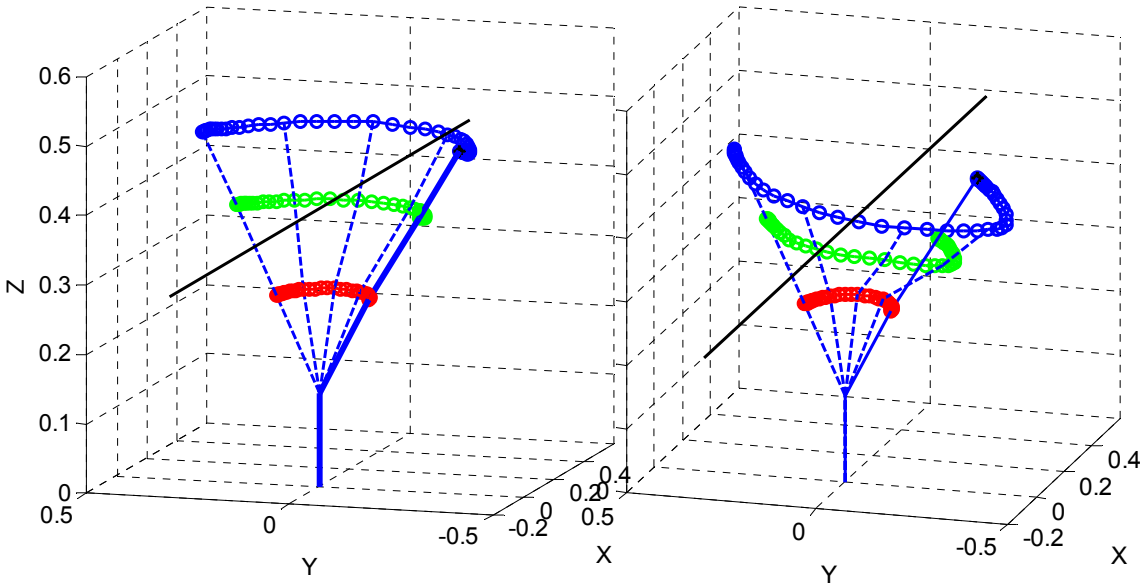

Figure 30. 3-DOF Maneuver Without and With Static Obstacle Avoidance: 60 Node Solution

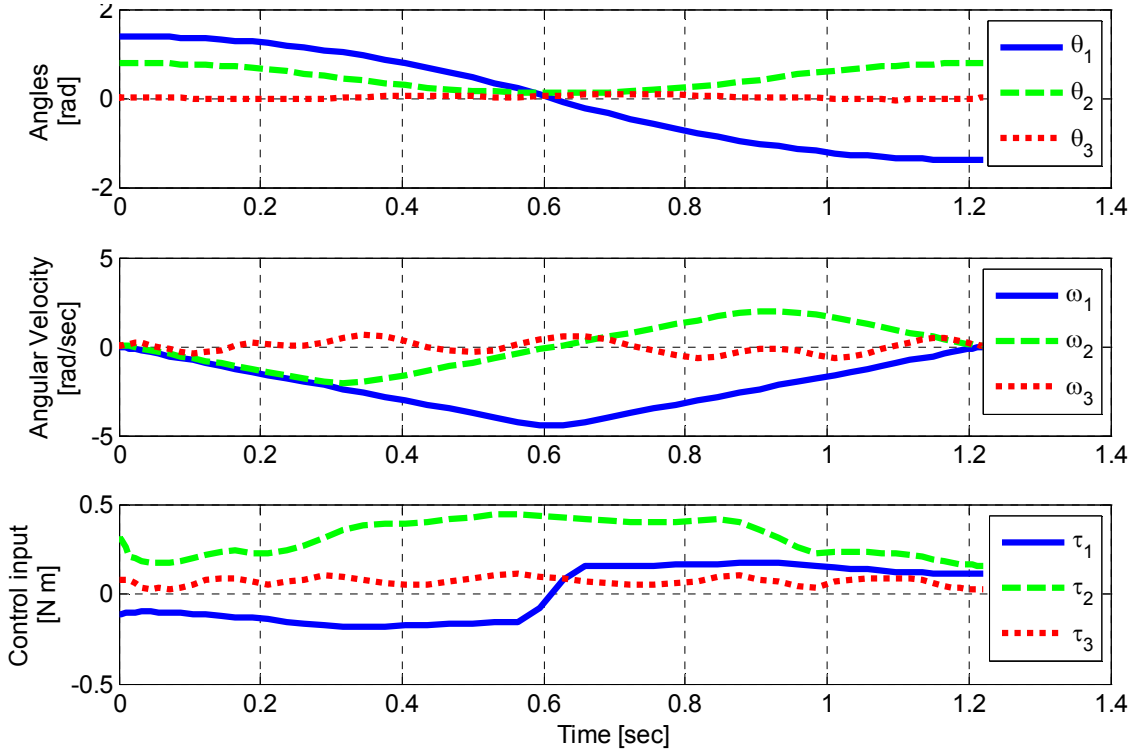

Figure 31. 3-DOF Motion with Static Obstacle Avoidance: State and Control Trajectories

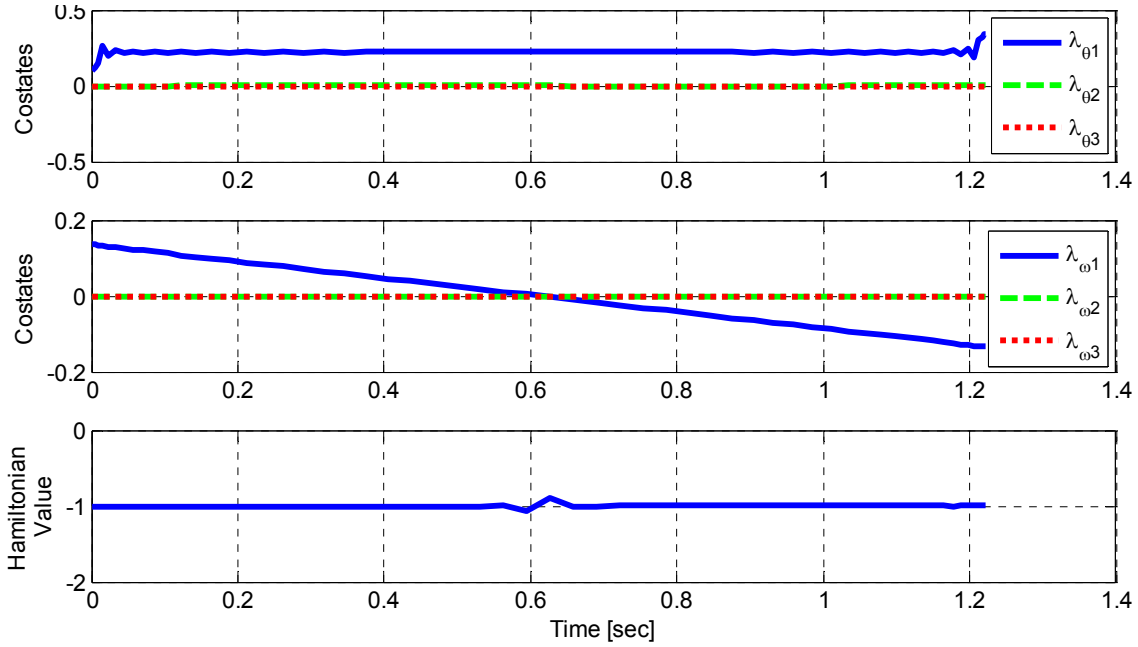

Figure 32. 3-DOF Motion with Static Obstacle Avoidance: Costates and Hamiltonian

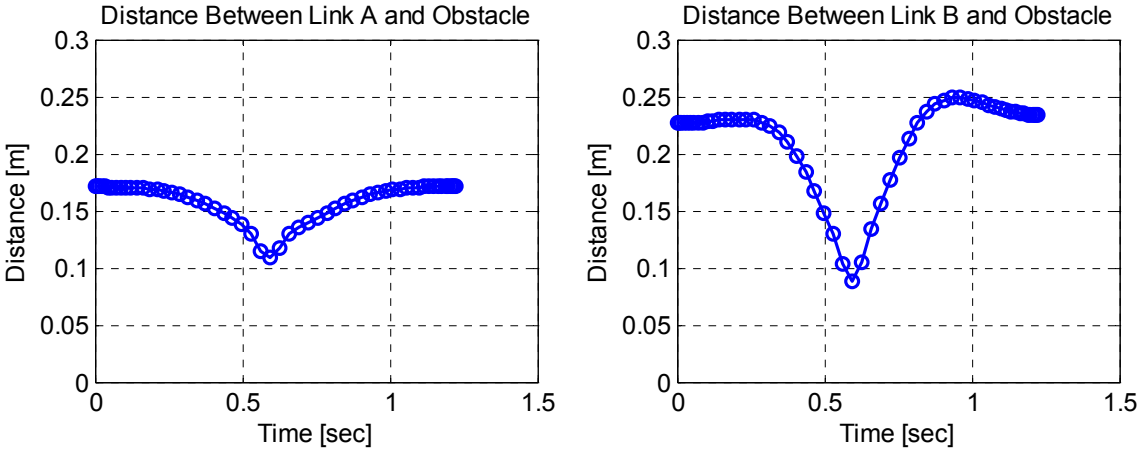

Figure 33. 3-DOF Motion with Static Obstacle Avoidance: Distance between Links and Obstacle

In applying the PS based obstacle avoidance algorithm, one must note that the numerical solution, uses discrete times or nodes to calculate the state and control trajectories and the path constraints are only tested and guaranteed at each of these nodes. A low node solution, while feasible at each discrete

node, may not to be physically realizable. Figure 34 and Figure 35 illustrate a 16- node solution for the same problem. The nodal spacing in these cases is such that the trajectory seems to jump the obstacle, that is the path constraints are met at the two nodes on either side of the obstacle as shown in Figure 35 but the arm trajectory would collide with the obstacle as it proceeds from one node to the next. The possibility of nodal jumping requires that the computed trajectory be analyzed to ensure it is feasible and may necessitate increasing the number of nodes, and therefore, the computational time to find a feasible solution.

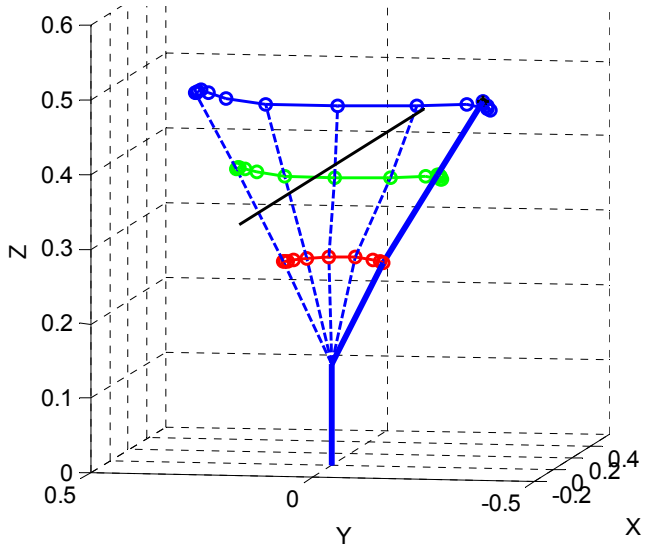

Figure 34. 3-DOF Motion with Static Obstacle Avoidance: 16 Node Solution

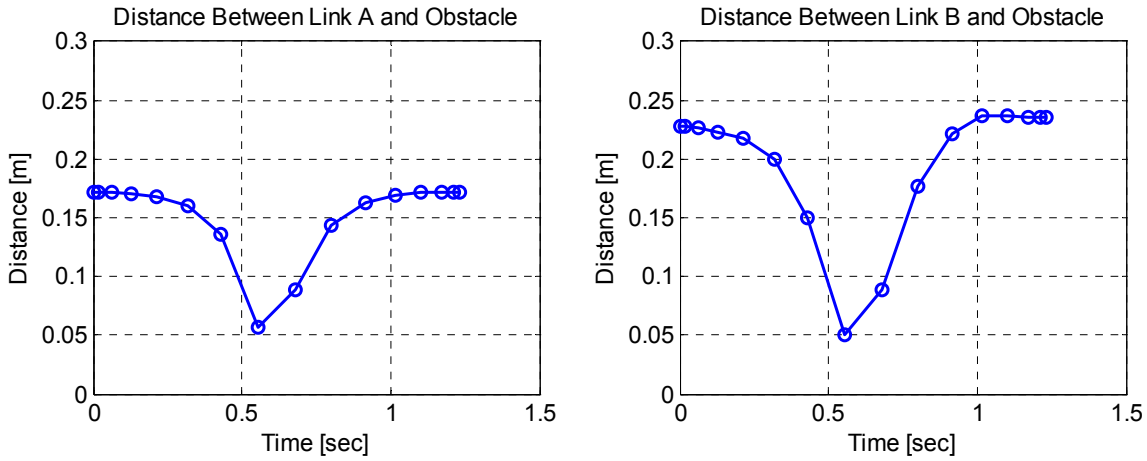

Figure 35. Minimum Distance Between Links and Obstacle,16 Node Solution

C. NUMERICALLY SOLVING THE KKT CONDITIONS

Returning to the discussion of minimum distance between line segments, solving Equation (41) should produce the minimum distance parametric coordinates. While this equation cannot be solved analytically, it can be solved numerically using DIDO.

Taking a step back to basic optimality, the constraints on the variables can be rewritten in the form $\mathbf{y}(\mathbf{x}) \leq 0$ where $\mathbf{y}$ is a vector of constraint functions.

$$\begin{aligned} \text{minimize} \quad & Y(t,s) = At^2 + Bt - Cst + Ds + Es^2 + F \\ \text{subj to} \quad & -t \leq 0 \qquad -s \leq 0 \\ & t-1 \leq 0 \qquad s-1 \leq 0 \end{aligned}$$

Taking the Lagrangian of the cost function yields a new optimization problem.

$$\begin{aligned} \text{minimize} \quad & \bar{Y}(\mathbf{x},\boldsymbol{\lambda}) = At^2 + Bt - Cst + Ds + Es^2 + F - \lambda_{t1}t + \lambda_{t2}(t-1) - \lambda_{s1}s + \lambda_{s2}(s-1) \\ \text{subj to} \quad & \boldsymbol{\lambda} \geq 0 \\ & \boldsymbol{\lambda}^T \mathbf{y} = 0 \end{aligned}$$

Rewriting Equation (40) using the new $\bar{Y}$ gives

$$\left.\frac{\partial \bar{Y}}{\partial \mathbf{x}}\right|_{\min} = \begin{Bmatrix} \dfrac{\partial \bar{Y}}{\partial t} \\ \dfrac{\partial \bar{Y}}{\partial s} \end{Bmatrix} = \begin{Bmatrix} 2At + B - Cs - \lambda_{t1} + \lambda_{t2} \\ -Ct + D + 2Es - \lambda_{s1} + \lambda_{s2} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{43}$$

with the constraints

$$\begin{aligned} \lambda_{t1}(-t) = 0 \qquad & \lambda_{s1}(-s) = 0 \\ \lambda_{t2}(t-1) = 0 \qquad & \lambda_{s2}(s-1) = 0 \end{aligned} \tag{44}$$

By letting s, t, and λ_i be six dummy control variables for each possible collision in the problem formulation, DIDO can numerically solve the problem with the KKT conditions that resemble the distance between two lines. Defining the minimum distance allowed between two arms, Equations (34), (43) and (44)

specify seven path constraints that are associated with each potential collision. While this increases the number of variables and functions, it directly solves the original obstacle avoidance problem without geometric checks and resulting pitfalls described by Figure 25.

1. 2-DOF Case

The motion planning problem was reformulated to include the dummy control variables and additional path constraints for the case of two 2-DOF arms:

$$\mathbf{u} = \left[\tau_1^a, \tau_2^a, \tau_3^a, \tau_1^b, \tau_2^b, \tau_3^b, t, s, \lambda_{t1}, \lambda_{t2}, \lambda_{s1}, \lambda_{s2} \right]^T$$

$$\mathbf{h}^L \le \mathbf{h}\left[\mathbf{x}(\square), \mathbf{u}(\square)\right] \le \mathbf{h}^u$$

$$d_{\min}^2 \le At^2 + Bt - Cts + Ds + Es^2 + F \le \infty$$

$$0 \le 2At + B - Cs - \lambda_{t1} + \lambda_{t2} \le 0$$

$$0 \le -Ct + D + 2Es - \lambda_{s1} + \lambda_{s2} \le 0$$

$$0 \le (-t)\lambda_{t1} \le 0$$

$$0 \le (t-1)\lambda_{t2} \le 0$$

$$0 \le (-s)\lambda_{s1} \le 0$$

$$0 \le (s-1)\lambda_{s2} \le 0$$

Using DIDO, the optimal trajectory was computed by solving the distance between the arms at each node and bounding the dummy control variables:

$$0 \le t \le 1$$

$$0 \le s \le 1$$

$$0 \le \lambda_i \le \infty$$

The results of this formulation are compared to those found using the geometric algorithm and presented in Figure 36 and Figure 37. The state and control trajectories are nearly identical with t_f and cost differences are on the order of 10^{-6}. The only significant difference was the computational time. On average, the KKT algorithm converged to a solution for the 2-DOF problems nearly twice as fast as the geometric algorithm.

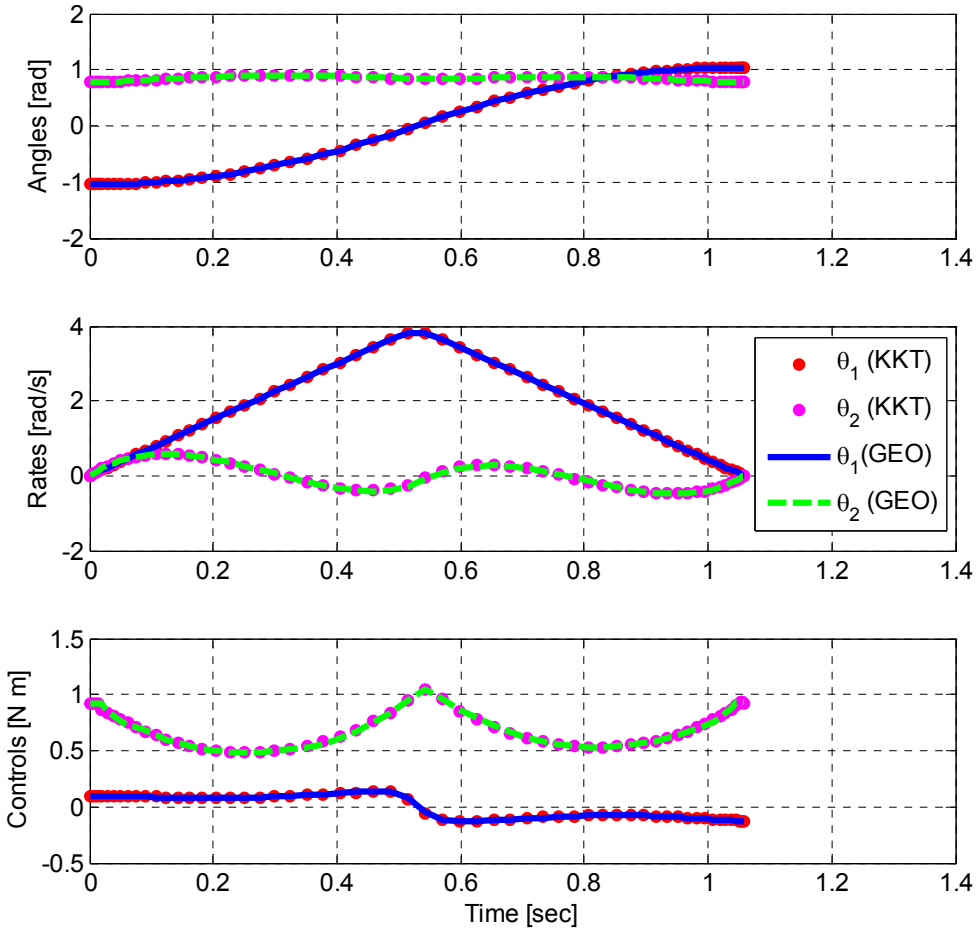

Figure 36. 2-DOF Static Obstacle Avoidance: State and Control Trajectories (KKT Algorithm versus Geometric Algorithm)

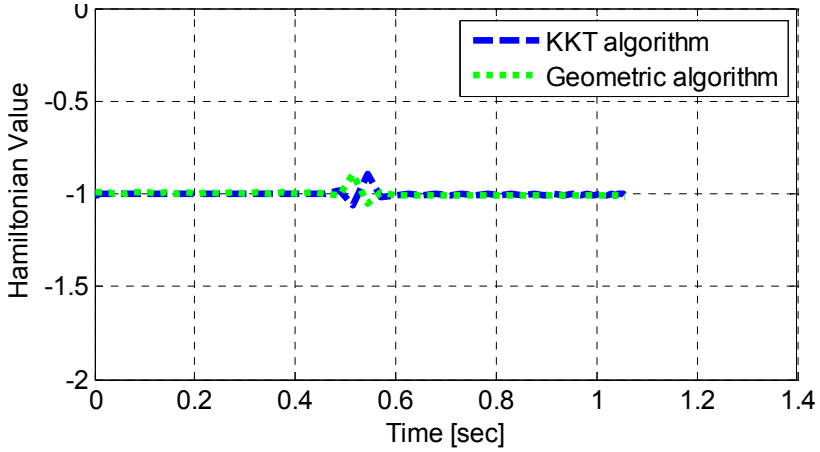

Figure 37. 2-DOF Static Obstacle Avoidance: Hamiltonian Values (KKT Algorithm versus Geometric Algorithm)

2. 3-DOF Case

The 3-DOF example was also reformulated using the KKT algorithm with an 18-element control vector, **u** and 14 path constraint equations. Figure 38 presents the results of this simulation. The maneuver time for both the geometric algorithm and the KKT algorithm are the same to within 10^{-3} at 1.221 seconds. Both functions used a cost function of the form

$$J = t_f + .1\int_0^{t_f} \mathbf{u}^T\mathbf{u} \tag{45}$$

The difference in cost between the two algorithms was 0.001 and may account for the slight difference in τ_2 and τ_3 trajectories. The Hamiltonian presented in Figure 39 corresponds to the required necessary condition, H=-1, just as it did for the geometric algorithm. The distance between the obstacle and the two links of the arm are presented in Figure 40.

The only major difference between the solutions for the two minimum distance algorithms is the computational time. It is difficult to accurately assess the runtime using DIDO because other processes are often running on the computer and solution times vary considerably for the same problem. Typically runtimes for a 16-node solution took 5-15 minutes regardless of the algorithm used. In an effort to make a meaningful comparison, a previous 60 node solution was used as an initial guess for each problem formulation. A 60 node solution took 115 seconds to run for the KKT algorithm compared to 93 seconds for the geometric algorithm. The trend appears that as the complexity of the problem increases and the dimension of control vector increases, the KKT solution is slower than the Geometric solution for the same problem.

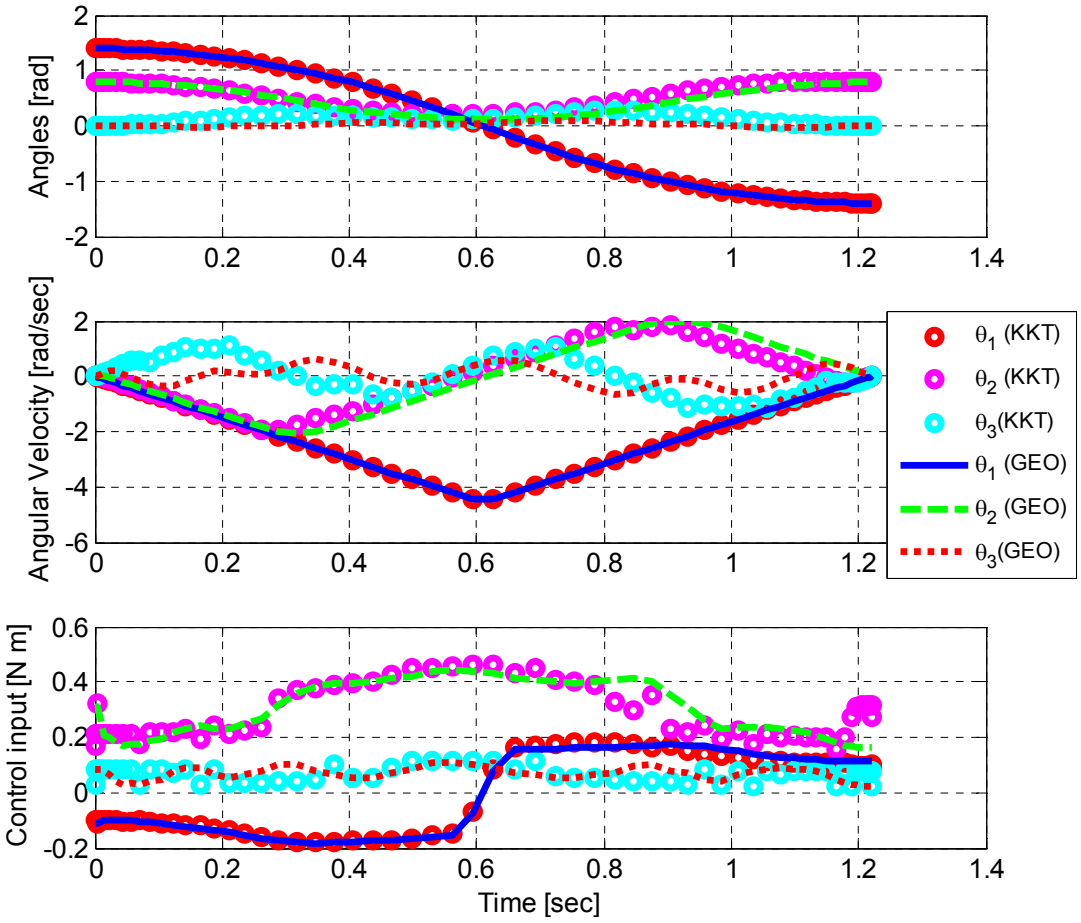

Figure 38. 3-DOF Static Obstacle Avoidance: State and Control Trajectories (KKT Algorithm versus Geometric Algorithm)

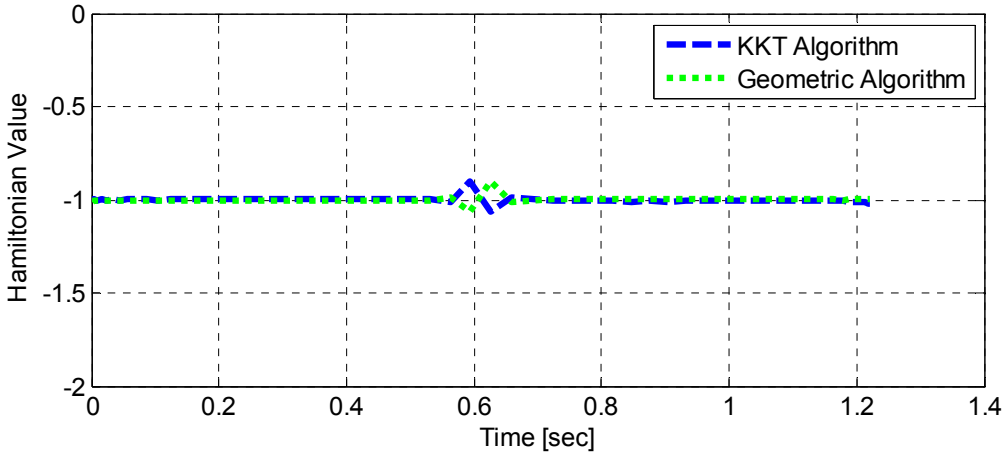

Figure 39. 3-DOF Static Obstacle Avoidance: Hamiltonian Values (KKT Algorithm versus Geometric Algorithm)

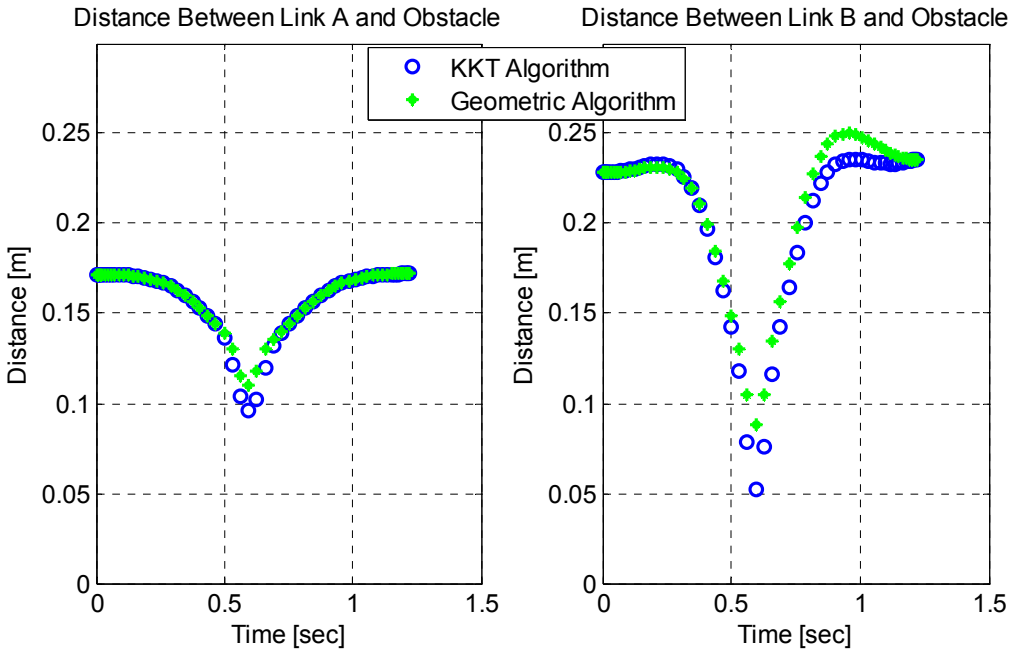

Figure 40. Minimum Distance Between Links and Obstacle (KKT Algorithm versus Geometric Algorithm)

THIS PAGE INTENTIONALLY LEFT BLANK

IV. COOPERATIVE PLANNING USING PS METHODS

The implementation of the obstacle avoidance algorithm in Chapter III demonstrates that pseudospectral techniques are capable of solving a range of problems. While the model was specific to an arm and a static cylinder-type object, the next step is to implement the same concept to find an optimal path to more complex systems. As part of this research, one interesting problem is to solve the optimal control trajectory of a multiple arm system where the arm links are modeled as line segments. Path planning for two arms has already been presented in Chapter II. Adding the path constraints from the obstacle avoidance algorithms presented in Chapter III allows for cooperative or, at the least, simultaneous control of a system of arms. Two examples are presented below using the dual arms that detail optimal path planning for 2-DOF and 3-DOF motions.

A. DUAL 2-DOF MANIPULATORS

The dual 2-DOF arm cooperative planning problem is a straight forward modification of the static obstacle formulation. The obstacle is now each link of the other arm. The minimum distance between the links of the two arms can be computed for any given time using either of the algorithms presented in Chapter III and can be constrained to be greater than some preset distance. Table 6 presents the endpoint conditions used to define the problem.

Arm a		Arm b	
$\theta_1^{a0} = 60^\circ$	$x_e^{af} = 0.337$	$\theta_1^{b0} = 0^\circ$	$x_e^{bf} = 0.102$
$\theta_2^{a0} = 60^\circ$	$y_e^{af} = -0.02$	$\theta_2^{b0} = 45^\circ$	$y_e^{bf} = 0.084$
	$z_e^{af} = 0.483$		$z_e^{bf} = 0.559$

Table 6. 2-DOF Cooperative Path Planning Endpoint Conditions

The two arms were offset by 0.30 m along the *y* axis and the final coordinates of the end-effector were chosen based on angles θ_1 = 0° and θ_2= 45° for Arm a and θ_1 = -60° and θ_2 = 60° for Arm b. DIDO was used to solve the problem with thecost function from Equation (45). The path constraint uses the endpoints of the arm links at each "node" to compute the distance between the arms and was constrained to be above some minimum distance d_{min}.

The optimal maneuver for this particular setup with no obstacle avoidance algorithm d_{min}= 0 has a closest point of approach of 10 cm and takes 0.75 seconds to complete. Figure 41 shows such unconstrained optimal state trajectory and Figure 42 is the minimum distance between the arms throughout the maneuver. For this maneuver, *J*=0.968.

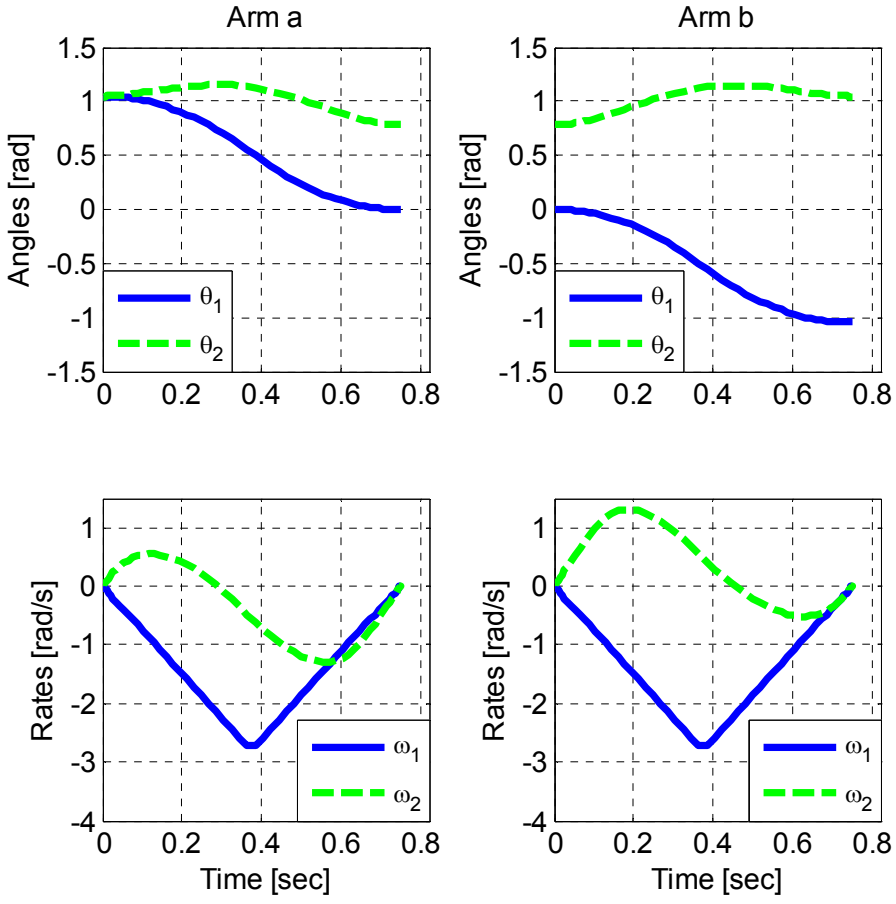

Figure 41. 2-DOF Cooperative Path Planning with No Obstacle Avoidance: State Trajectory

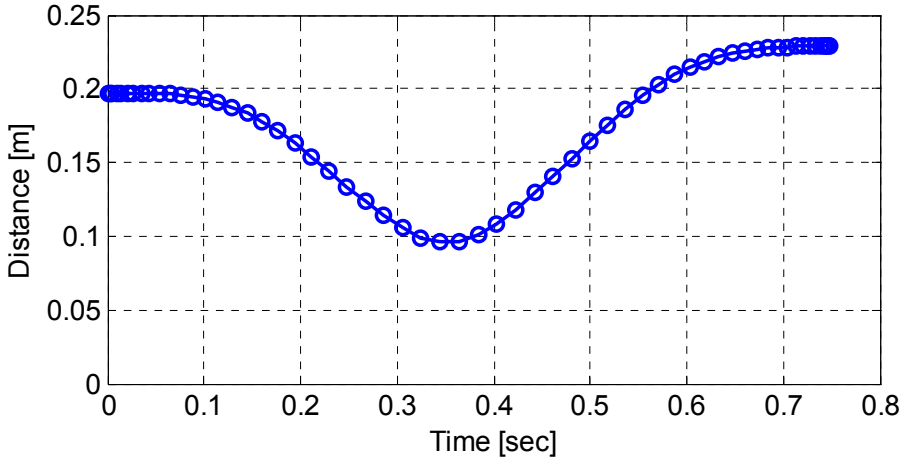

Figure 42. 2-DOF Cooperative Path Planning with No Obstacle Avoidance: Distance between Arms

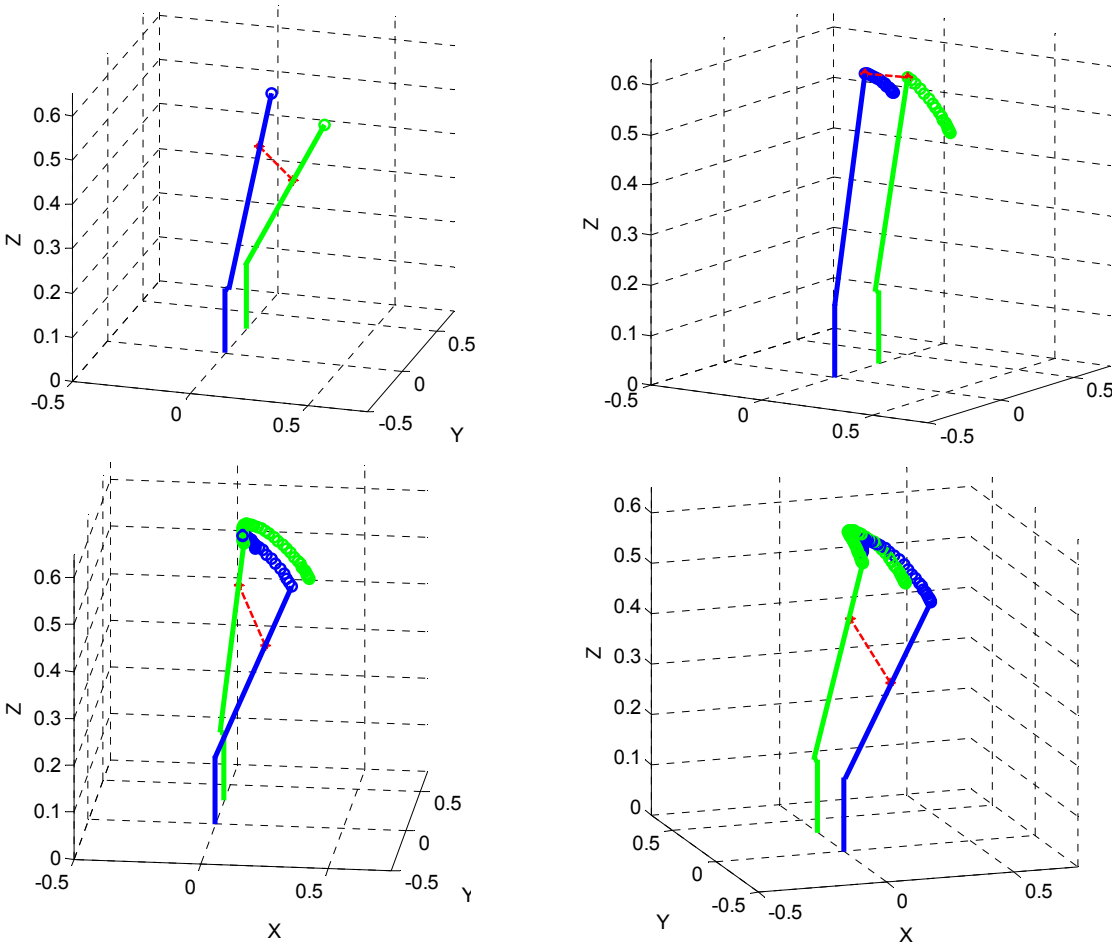

Figure 43. Cooperative Path Planning for Dual 2-DOF Arms with 19.5 cm Minimum Clearance

If 10 cm is not sufficient for the arms to avoid colliding it is imperative to modify the problem and define a more restrictive d_{min}. For the maneuver with the above endpoint conditions, the maximum buffer distance between the arm links is 19.7 cm. The following example uses the same endpoint conditions; however, d_{min} was given as 19.5 cm. DIDO was run using both the geometric and KKT collision avoidance algorithms with similar results. Figure 43 depicts the arms at four points along the computed path. A red line is attached to the points on each arm where the minimum distance between the arms occur. While not clear in the plots, that minimum distance line is perpendicular to both arms. Figure 44 - Figure 47 show results using the geometric algorithm for obstacle avoidance.

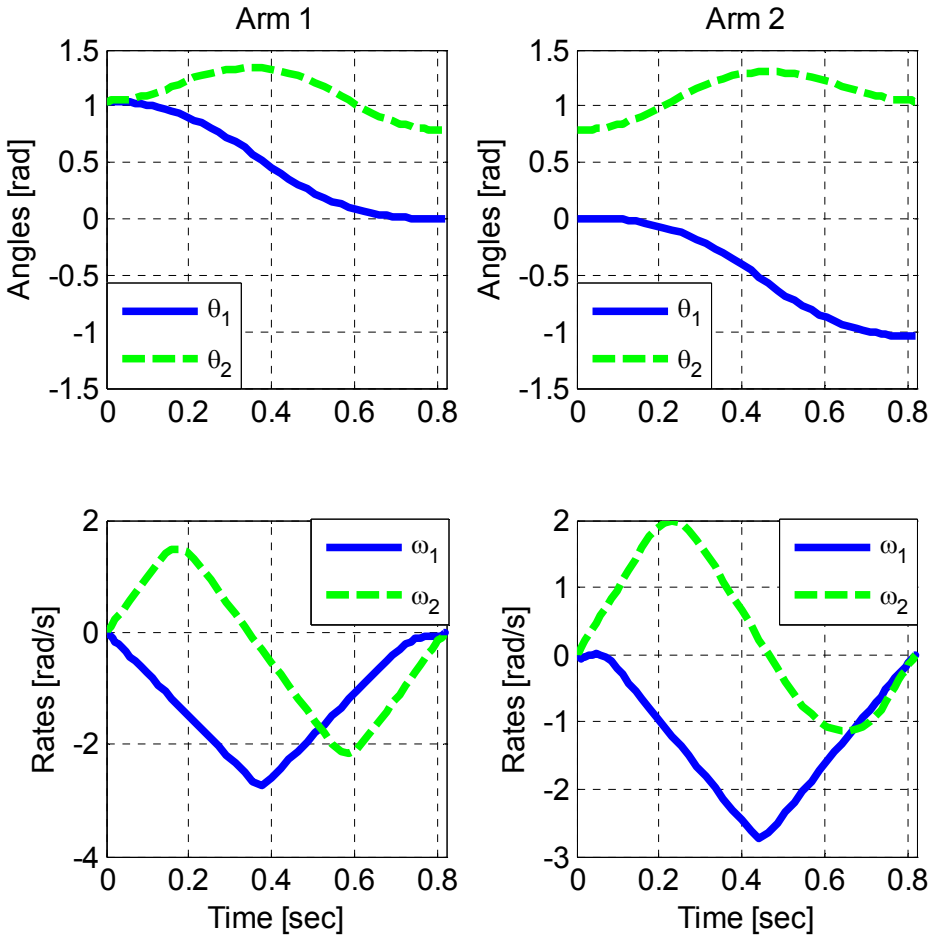

Figure 44. 2-DOF Cooperative Path Planning using Geometric Algorithm and 19.5 cm Buffer: State Trajectories

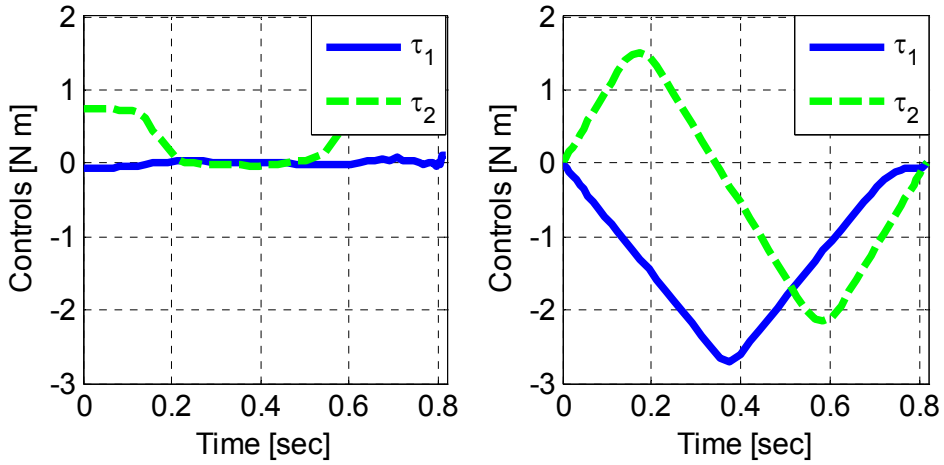

Figure 45. 2-DOF Cooperative Path Planning using Geometric Algorithm and 19.5 cm Buffer: Control Trajectories

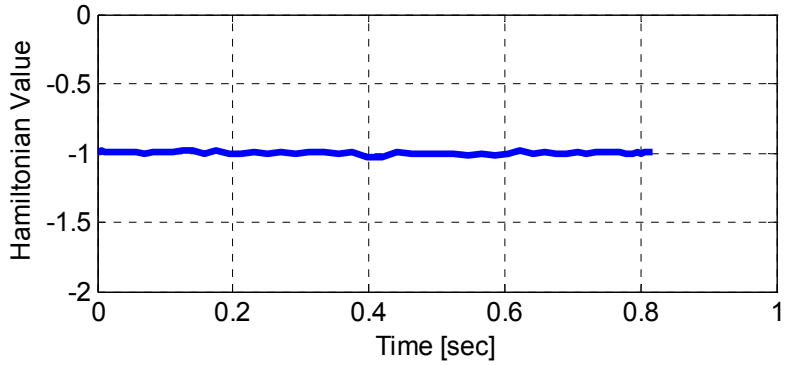

Figure 46. 2-DOF Cooperative Path Planning using Geometric Algorithm and 19.5 cm Buffer: Hamiltonian Value

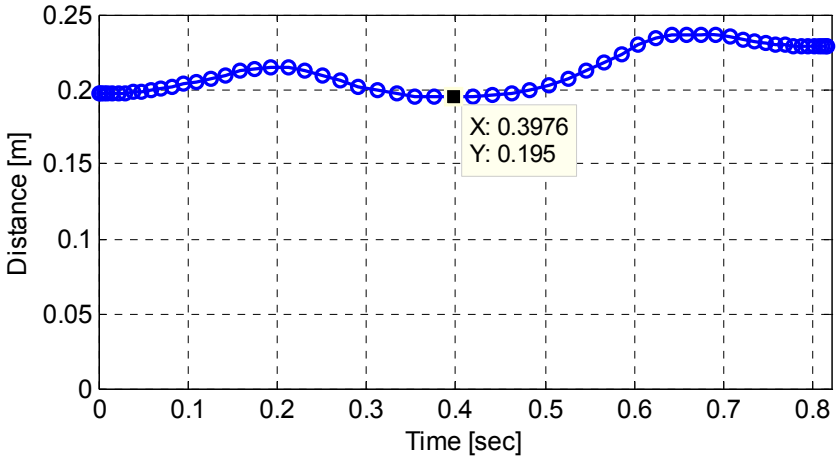

Figure 47. 2-DOF Cooperative Path Planning using Geometric Algorithm and 19.5 cm Buffer: Minimum Distance between Arms

Compared to the case with no obstacle avoidance, the total cost of the dynamic optimization problem for the optimal path maneuver including a 19.5 cm buffer was found to be J=1.067, a 10% increase. The maneuver time also increased to 0.82 seconds and the weighted quadratic cost increased by .02 units. The only substantial difference between the KKT algorithm and the geometric algorithm was the computational time to converge to a solution. On average, the KKT formulation was greater than twice as fast as the geometric algorithm. Using a 60-node solution with a previously computed 60-node guess, the KKT algorithm took 11.9 seconds to solve versus 35 seconds for the geometric algorithm.

B. DUAL 3-DOF MANIPULATORS

The multiple 3-DOF robotic arm obstacle avoidance problems are more complicated. The multiple time-optimal paths for each arm plus the nearly infinite obstacle avoidance paths make this an interesting, but challenging problem. While the algorithms presented to this point are theoretically capable of solving the problem, the number of potential collisions create more complex path functions. For this 3-DOF, dual arm system, there are six potential collisions that must be accounted for vice one in the relatively simple 2-DOF system above. In the case of the KKT algorithm, this results in 42 path constraints and 36 "dummy" control variables that compute the distance between any two links. Table 7 summarizes the endpoint constraints used for this simulation.

Arm A		Arm B	
$\theta_1^{a0} = -45^\circ$	$x_e^{af} = 0.435$	$\theta_1^{b0} = 90^\circ$	$x_e^{bf} = 0.435$
$\theta_2^{a0} = 45^\circ$	$y_e^{af} = -0.030$	$\theta_2^{b0} = 45^\circ$	$y_e^{bf} = -0.330$
$\theta_3^{a0} = 45^\circ$	$z_e^{af} = 0.270$	$\theta_3^{b0} = 0^\circ$	$z_e^{bf} = 0.270$

Table 7. 3-DOF Cooperative Path Planning Endpoint Conditions

For this example, $\theta_i^f = 0^\circ$ was used to compute the final endpoint position in Table 7 for both arms to ensure the feasibility of the final condition. Arm B was placed 30 cm from Arm A in the –y direction with the same 3-DOF limits from Equation set (25). As was presented in the 2-DOF case, Figure 48 and Figure 49 present the results with d_{min}=0. The closest approach occurred between the manipulator links of the two arms (Link 3 in Figure 49) when they come within 1.6 cm.

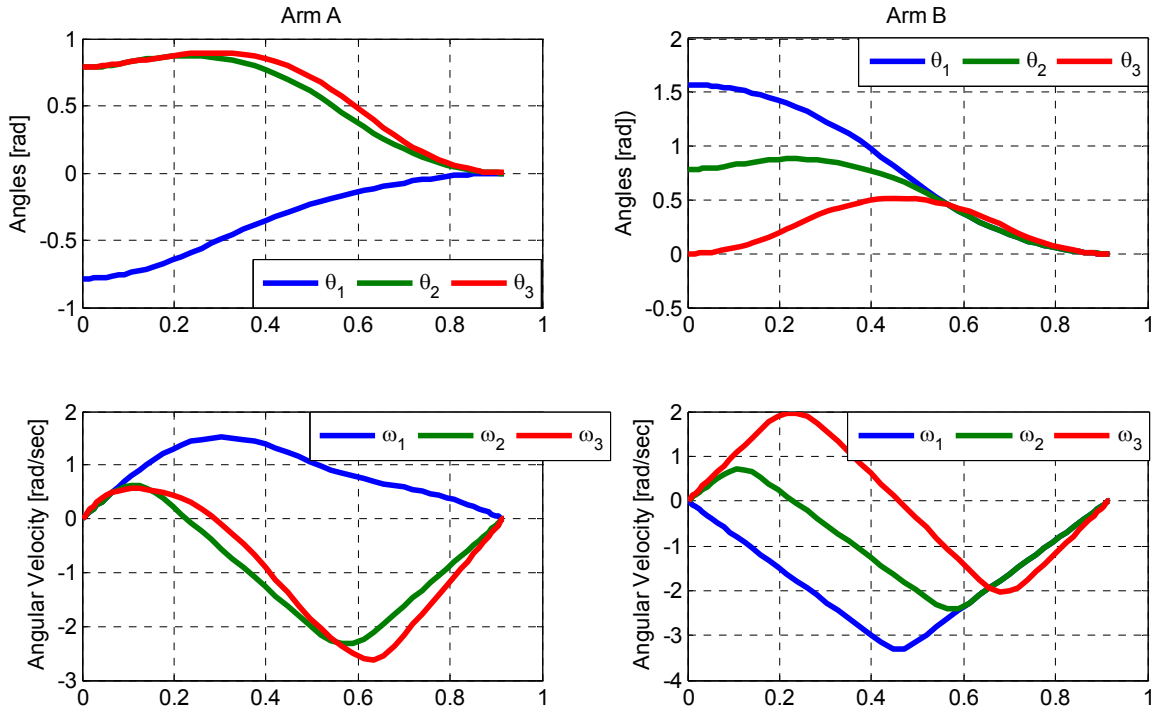

Figure 48. 3-DOF Path Planning without Obstacle Avoidance: State Trajectories

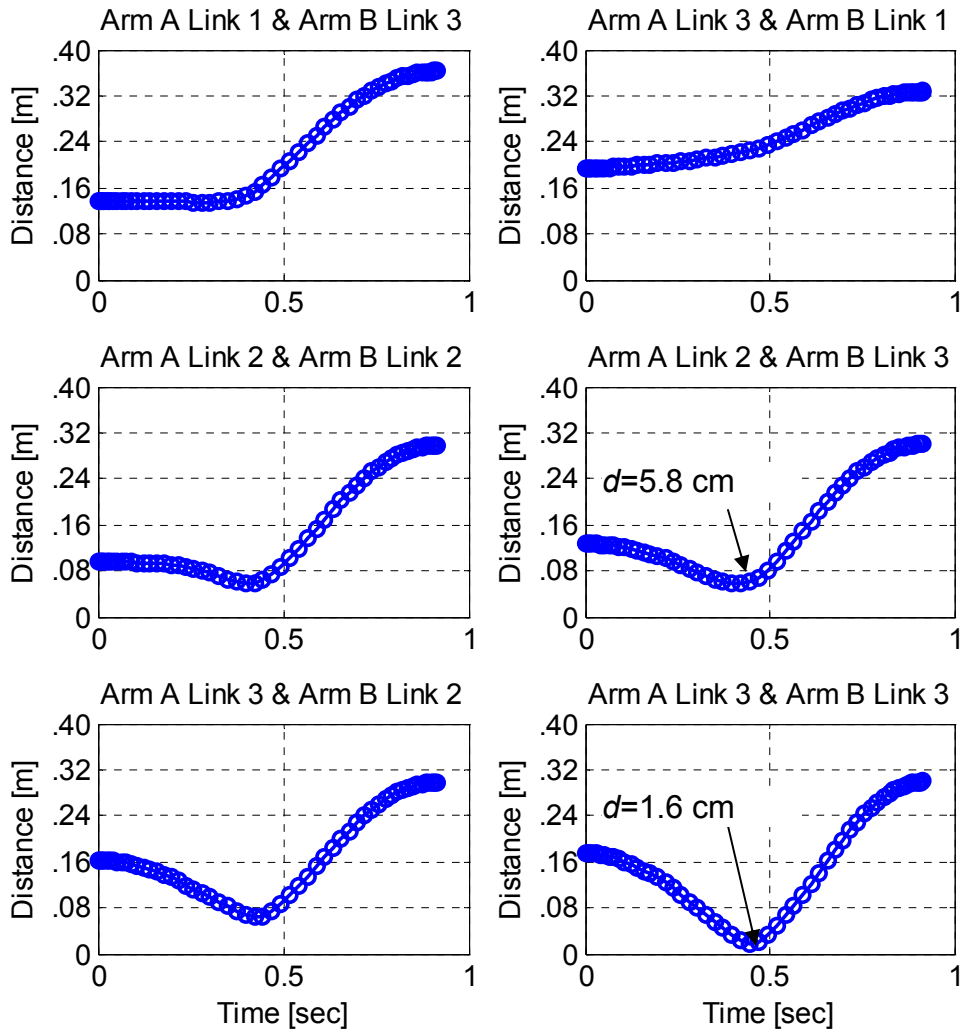

Figure 49. 3-DOF Path Planning without Obstacle Avoidance: Distance between Links

One of the advantages of using DIDO is the ease in which the problem can be designed. For example, the buffer distance can be tailored for each potential collision based on the physical dimensions of each link. In this case, each link was assumed to be the same width and d_{min}= 8 cm for all six potential collisions. DIDO was run using an initial three-point guess for a 16-node solution and then a bootstrap approach for subsequent iterations. The middle point of this guess was nominally chosen so that both arms would orient vertically with θ_1 at the midpoint the same as θ_{10}. Intuitively, raising the arms to vertical before the

base turns would create separation and provide the arms with an obstacle free path in most cases. In general, this was found to reduce the computation time. Figure 50 sketches the 60-node optimal path of both arms computed by DIDO with an d_{min} = 8 cm. The obstacle avoidance maneuver took the same time as the the case with d_{min} = 0 cm, 10^{-8} seconds, with an increase in the quadratic cost of 3.5%. Figure 51 -Figure 53 display the state and control trajectories, costate values, and Hamiltonian values computed by DIDO.

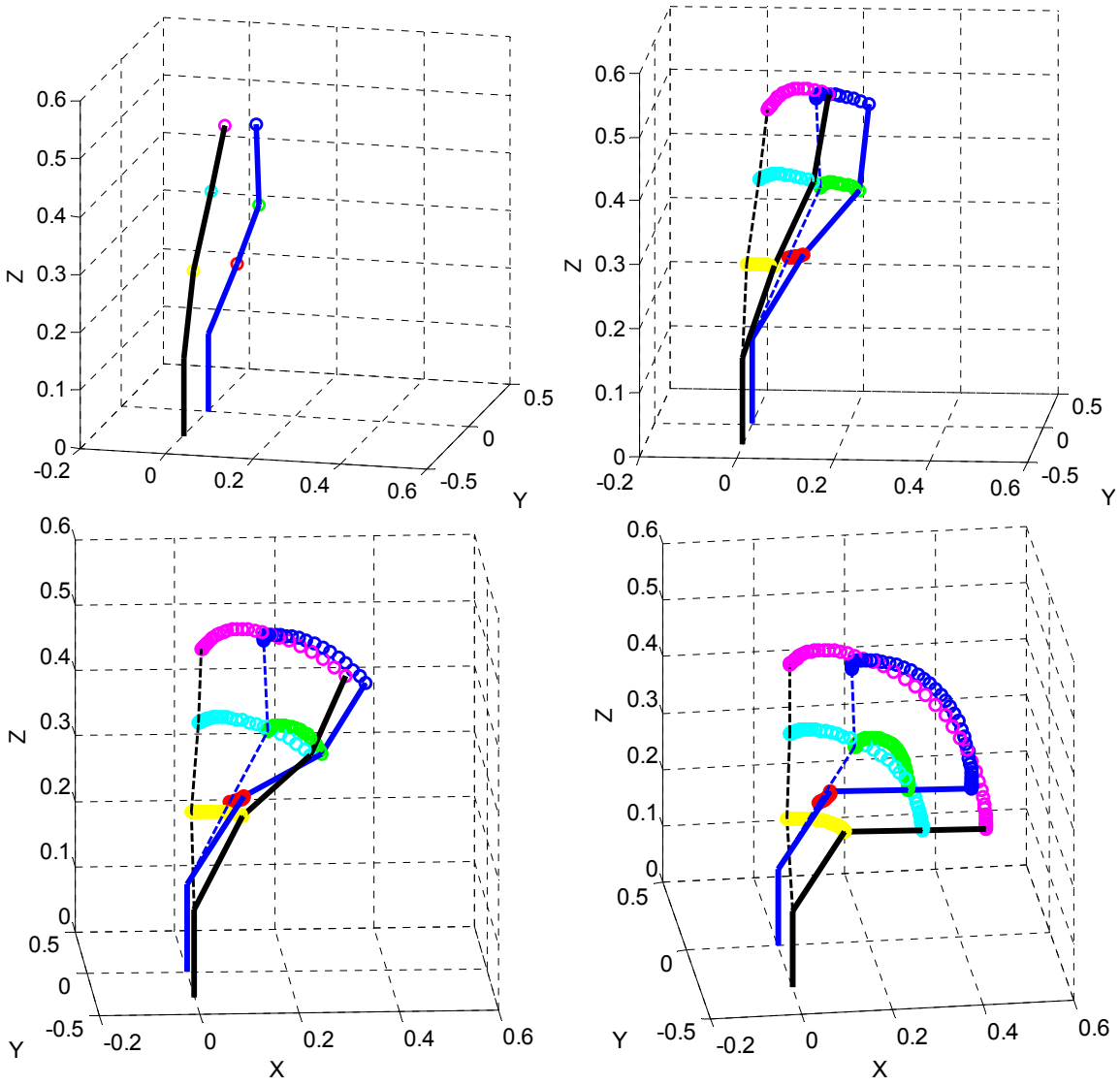

Figure 50. Cooperative Path Planning for Dual 3-DOF Arms with 8 cm Buffer

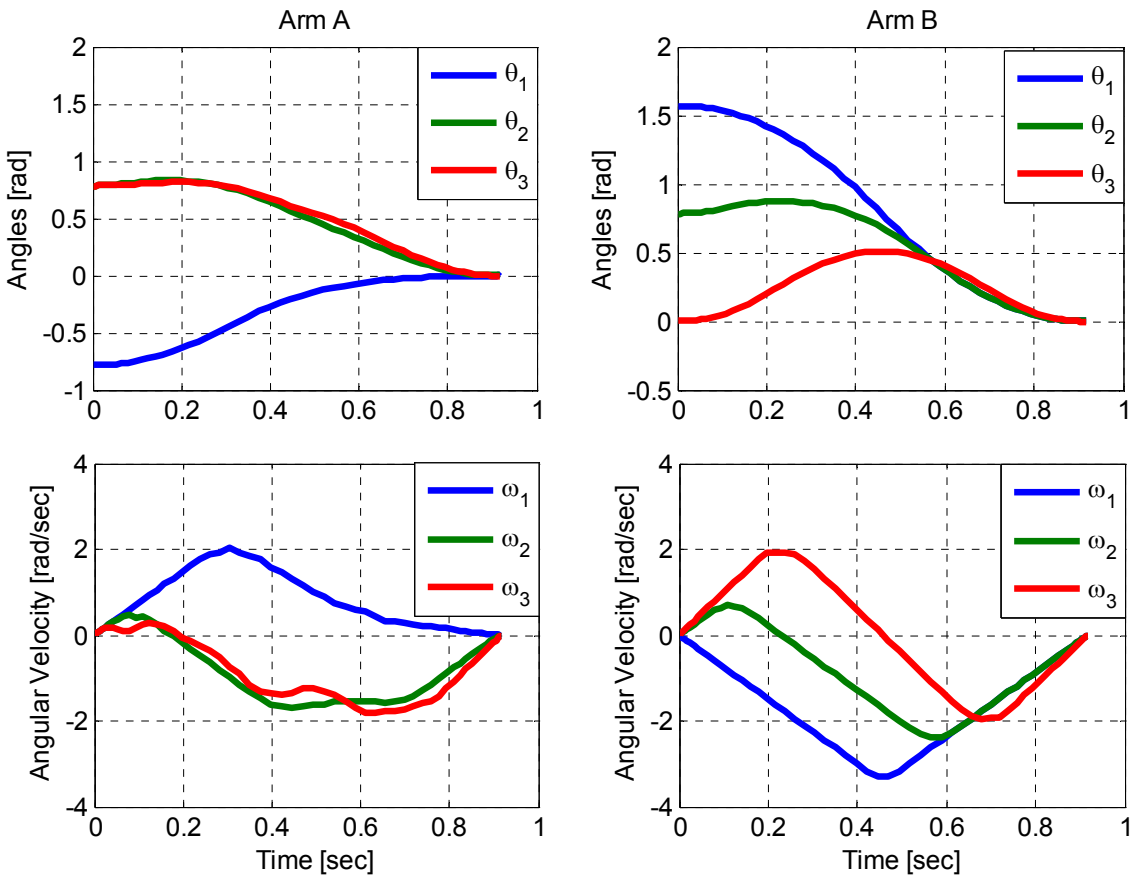

Figure 51. 3-DOF Cooperative Path Planning using Geometric Algorithm and 8 cm Buffer: State Trajectories

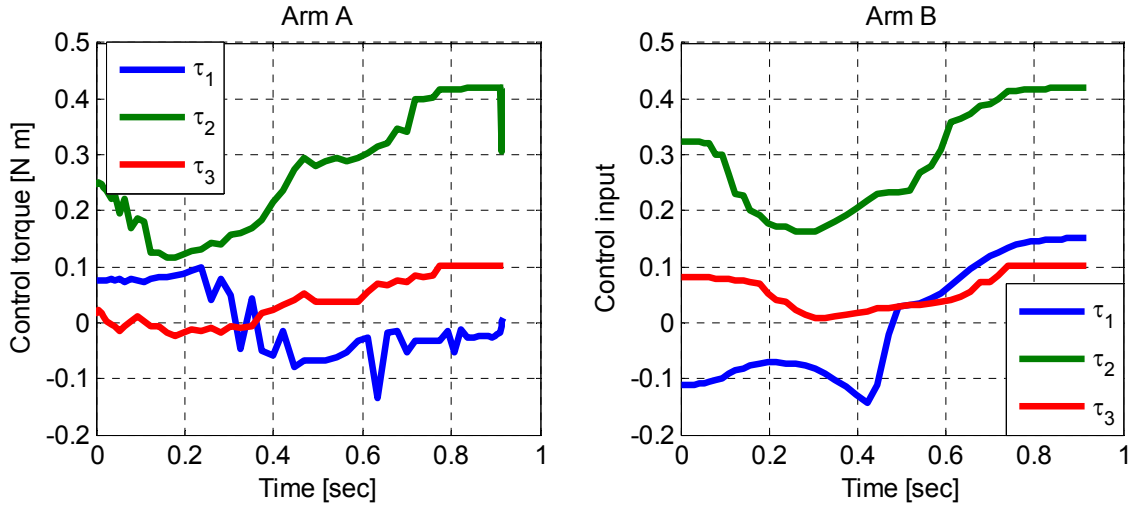

Figure 52. 3-DOF Cooperative Path Planning using Geometric Algorithm and 8 cm Buffer: Control Trajectories

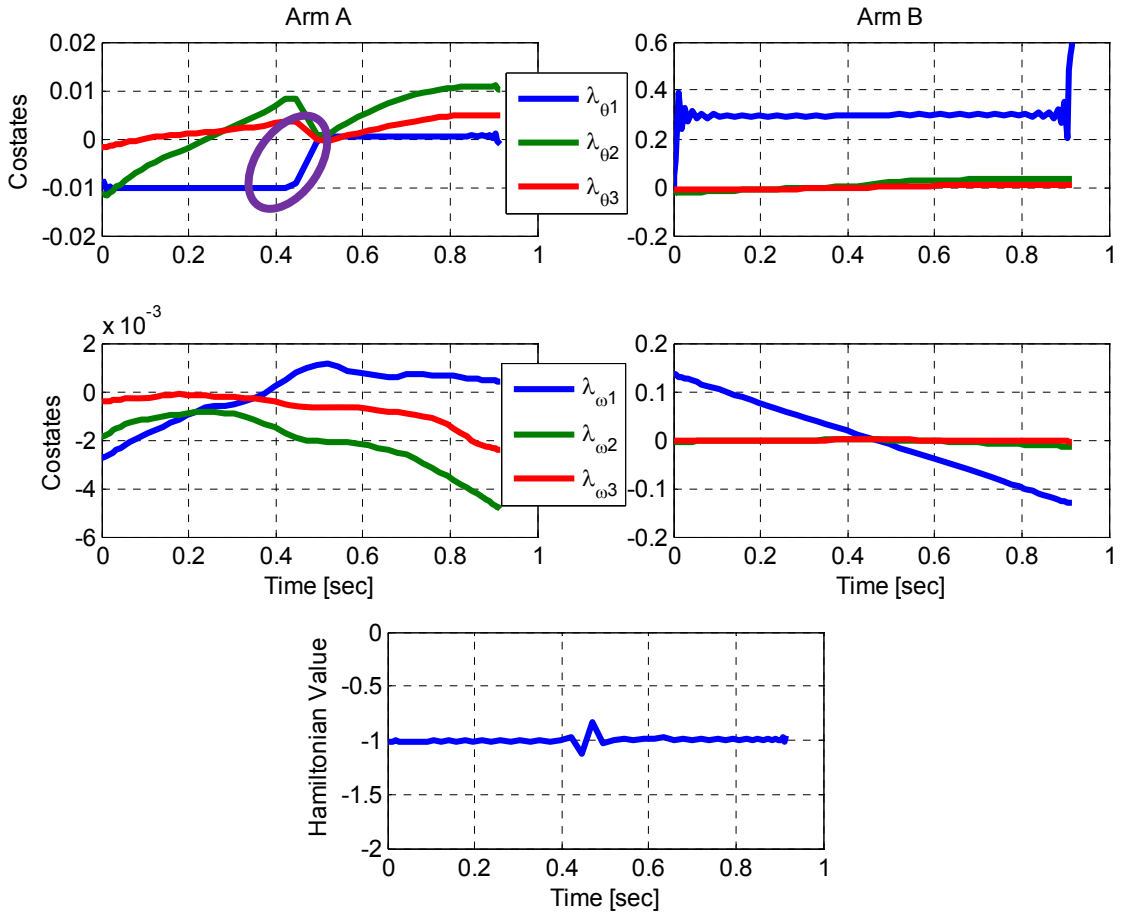

Figure 53. 3-DOF Cooperative Path Planning using Geometric Algorithm and 8 cm Buffer: Costate and Hamiltonian Values

While the Hamiltonian value is constant at -1 from Figure 53, the value for $\lambda_{\theta 1}$ for Arm A does not appear to be constant and steps up at the midpoint of the maneuver. Changing the scale of the chart and plotting the Arm B $\lambda_{\theta 1}$ values on top of the Arm A $\lambda_{\theta 1}$ shows that this step up is of the same order of magnitude as the oscillations in the other costate values due to numerical computation errors seen in Figure 54. Figure 55 illustrates that the avoidance algorithm does provide the required buffer of 8 cm between the links.

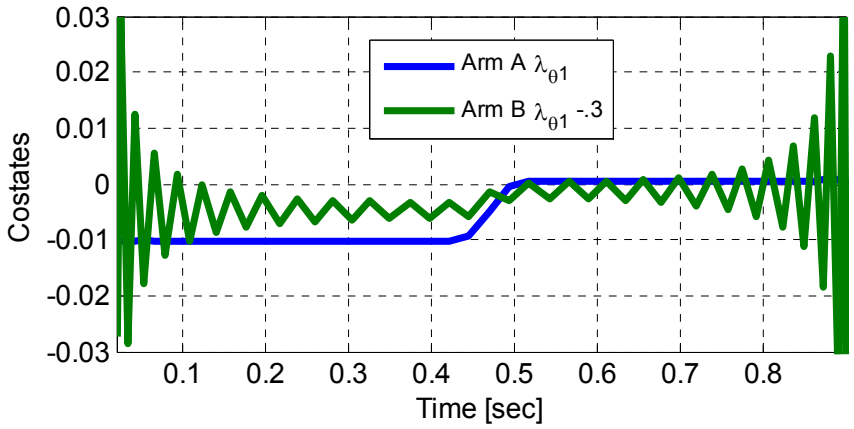

Figure 54. 3-DOF Cooperative Path Planning using Geometric Algorithm and 8 cm Buffer: $\lambda_{\theta 1}$ Plot

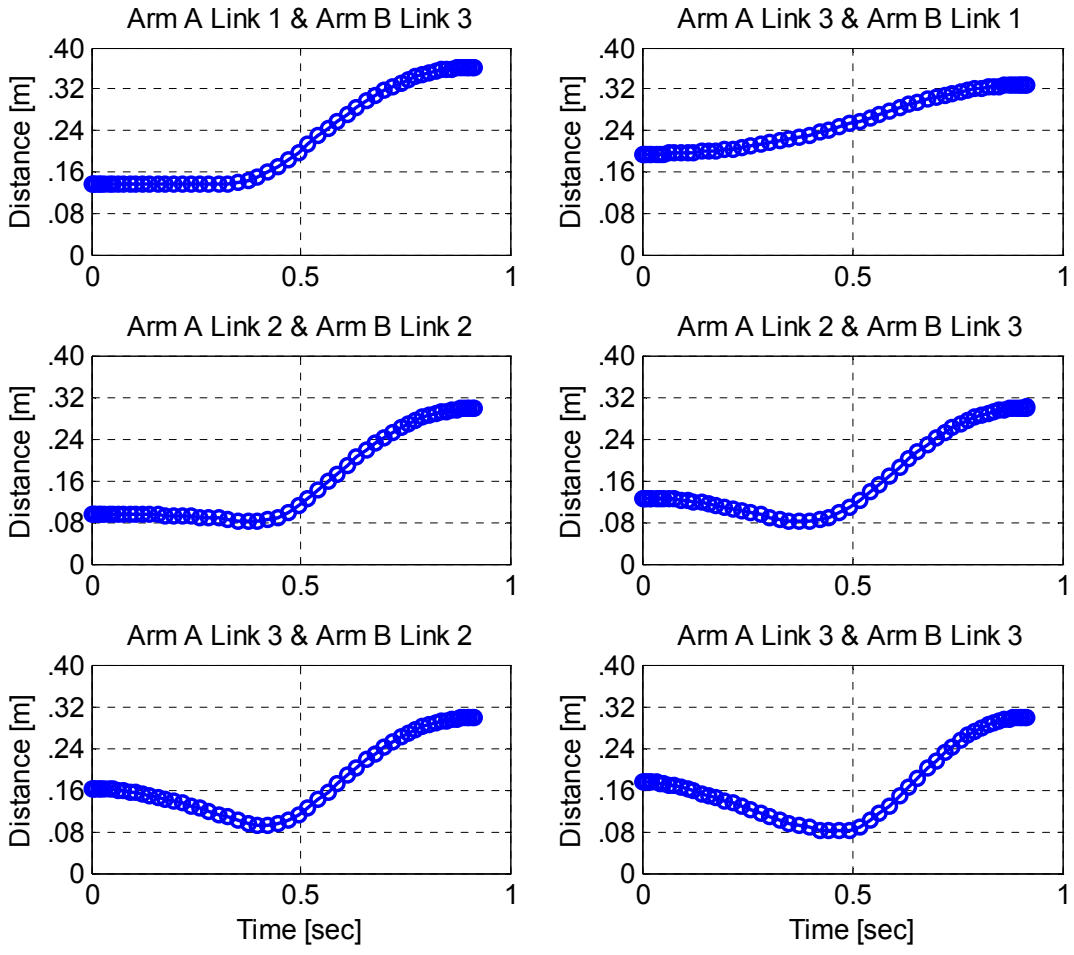

Figure 55. 3-DOF Cooperative Path Planning using Geometric Algorithm and 8 cm Buffer: Distance between Arms

To ensure feasibility of the solution, the DIDO computed control trajectory was linearly interpolated and used to propagate the system using Matlab's ODE45 function and a fixed time step of 0.05 seconds. The results of the propagated states, shown in Figure 56, verify the feasibility of the control solution. The KKT algorithm was also run using the endpoint constraints from Table 7 with nearly identical results to the geometric algorithm. Figure 57 compares the two state trajectories and Figure 58 plots the costate values that correspond well with the geometric algorithm shown in Figure 53.

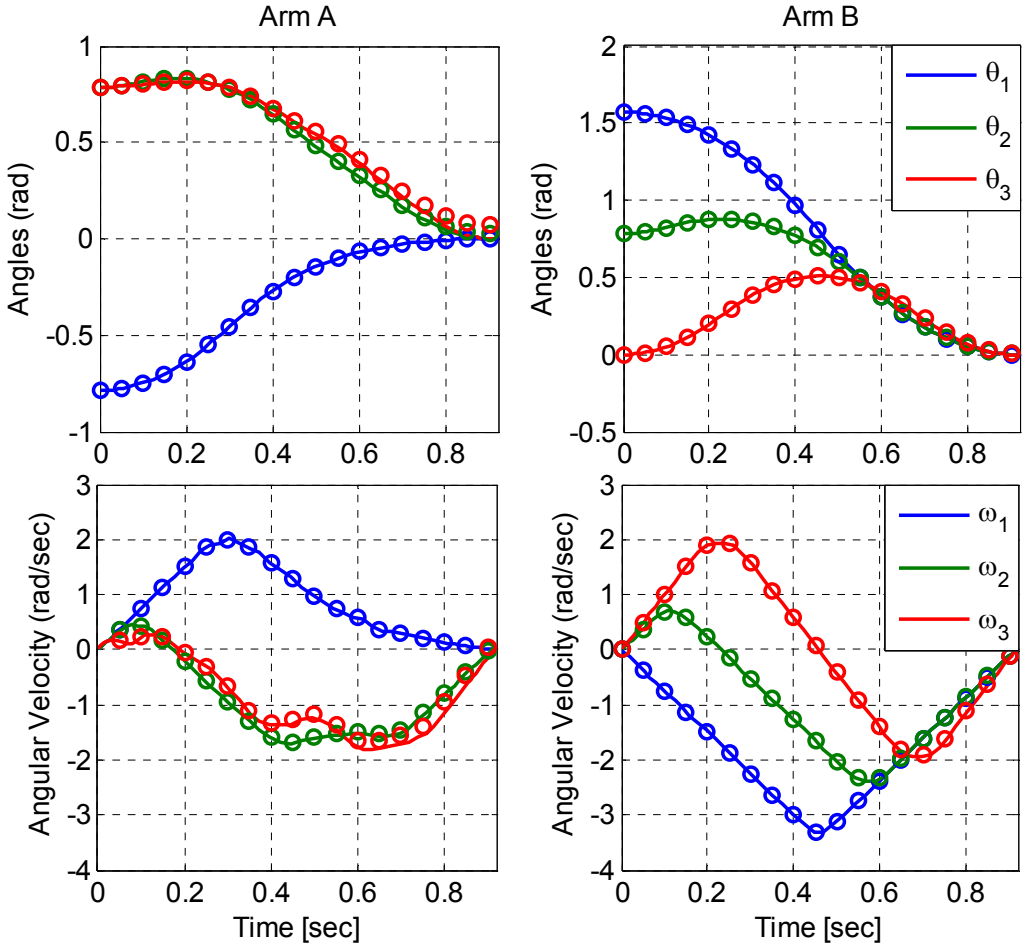

Figure 56. 3-DOF Cooperative Path Planning using Geometric Algorithm and 8 cm Buffer: Propagated State Values (o) compared with DIDO values (-)

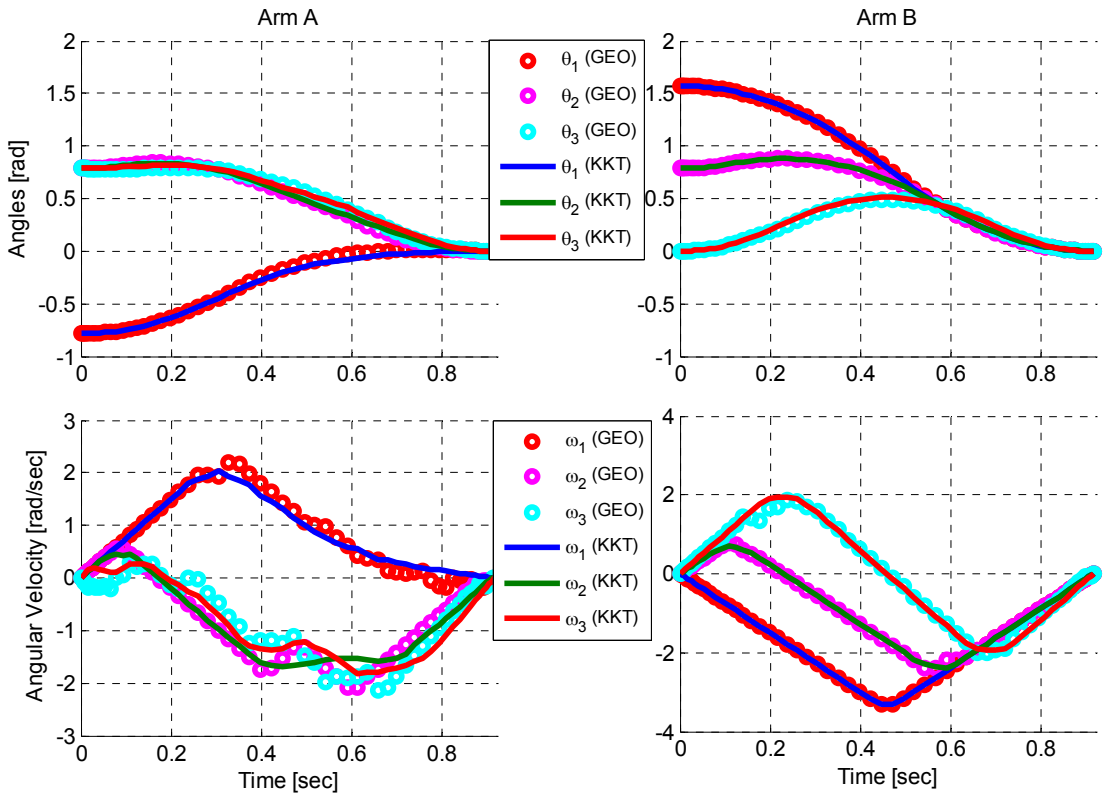

Figure 57. 3-DOF Cooperative Path Planning: State Trajectory (KKT Algorithm versus Geometric Algorithm)

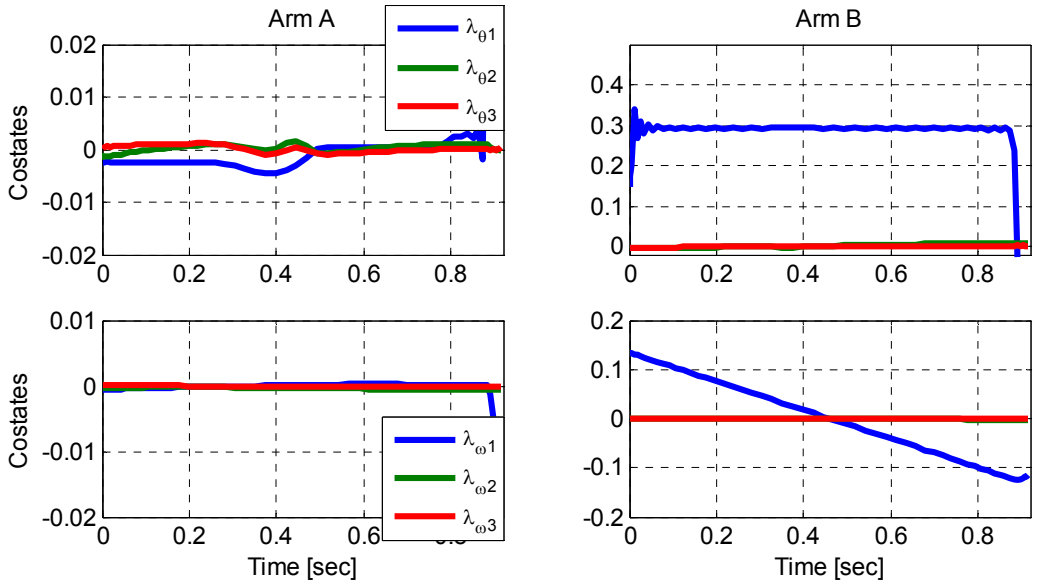

Figure 58. 3-DOF Cooperative Path Planning using KKT Algorithm: Costate Values

While the solution paths between the KKT and geometric algorithms are similar, two differences are worth pointing out. Using the KKT algorithm, DIDO computed a cost function that was 7% lower than when it used the geometric algorithm. Since the final maneuver time was the same to 10^{-8} seconds, the difference corresponds to a 21% decrease in the quadratic cost. While the KKT algorithm seems to converge to a better cost function, a computational price is paid. On average DIDO took three times longer to converge to a solution using the KKT algorithm than using the geometric algorithm. The fact that the KKT algorithm converges to a slightly lower cost solution infers that the geometric algorithm may not be as accurate for complex systems and the KKT algorithm may be more robust.

Whether this computational penalty can be alleviated is an interesting path for future work. In general, with the 3-DOF problems visited in this work, the costate values were an order of magnitude smaller than the corresponding state values. This brief comparison may indicate that the problem can be better scaled and balanced using some unknown designer units, which would allow the pseudospectral method employed by DIDO to converge faster to a more accurate optimal solution.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND RECOMMENDATIONS

Application of pseudospectral methods for motion planning of multiple mulit-DOF robotic manipulators was studied. The use of DIDO for obtaining optimal trajectories allowed the focus of effort to be placed on the problem formulation instead of solving the optimal control problem. Rather than starting with a complex problem that incorporates trajectory planning and obstacle avoidance, a staged approach was found to be an effective means of developing the final optimal control problem formulation. First, a relatively simple 2-DOF problem was formulated and solved with all the necessary conditions for optimality derived and verified. Building on this basic problem formulation, the complexity of the system was increased to 3-DOF and then with the addition of a second robotic manipulator a 6-DOF system. Pseudospectral techniques were effective in quickly solving optimal pick-and-place paths. While not trivial, the addition of higher DOF arms and increasing the number of arms can be accomplished relatively simply within the pseudospectral framework.

The choice of cost function is an important element that has a major effect on the ultimate solution. While a minimum time trajectory is desired, some level of efficiency should be included, particularly when dealing with decoupled elements, or even lightly coupled elements as is often the case with multiple links and arms. A weighted cost function that includes both a minimum time element and a measure of minimizing effort is desirable. Once again, the use of DIDO allows the problem formulation to be flexible and easily accommodate a wide range of cost functions based on the mission objectives.

Even the relatively simple model of a robotic arm using rigid links and perfect joints had a fairly complex form. A higher fidelity model that includes joint parameters and system flexibility can be substituted in for the dynamics of the system. While the level of effort to model the system would surely increase, the optimal control framework may exploit the more complex interactions to find a solution that increases the performance of the system.

Obstacle avoidance was included by defining path constraints that consisted of the minimum distance between the manipulators' links. Using parametric equations to define each link, an optimization problem was formulated and simultaneously solved to determine the minimum distance between each link and any potential obstacle. Two techniques are presented that take advantage of the flexibility of problem formulation in DIDO. The results suggest that for higher order systems, the geometric function for finding the minimum distance converges faster when provided a feasible guess. The algorithm based on numerically solving the KKT conditions seems to be better behaved and solves a low-node problem with a simple two-point guess more reliably. As the complexity of the system increases, the robustness of the KKT algorithm can be used to solve a low node solution, which can then be used as a guess for the geometric algorithm to refine the trajectory with less computation time. The refined control solution can then be interpolated and used for a physical implementation.

As implemented, the use of the proposed obstacle avoidance techniques requires knowledge of all obstacles in the workspace. It also assumes that each link in the system is modeled as a line, which may not be ideal for some systems. Further efforts can be placed into modifying the parametric distance function for complex shapes.

By not having to focus on an analytical solution to the optimal control problem, variations in problem formulation can be explored. Being able to compare the results of different problem formulations may result in the discovery of computational efficiencies. Here, no scaling was done on any variables. By experimenting with designer units, the computation time to solve the problem can be improved with no change in the physical trajectory. Another variation not considered in this study is the use of the links' minimum distance function in the cost as a pseudo-repulsive force or penalty function.

While the robotic manipulators studied here are relatively simple, the techniques presented can be used to solve more complicated problems that just a few years ago were considered unsuitable for real-time use [5]. Collision

avoidance and trajectory planning for cooperating arms using pseudospectral methods can also be studied for real-time autonomous implementation provided computational efficiencies are exploited.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX - 2-DOF ARM DYNAMICS COMPUTATION

```
**************************************************
Step 1: Construct the DH transformation matirx and its inverse...
**************************************************
> H := Matrix([[cos(θ), -cos(α)·sin(θ), sin(α)·sin(θ), a·cos(θ)], [sin(θ), cos(α)
      ·cos(θ), -sin(α)·cos(θ), a·sin(θ)], [0, sin(α), cos(α), d], [0, 0, 0, 1]]);
  Hinv := Matrix([[cos(θ), sin(θ), 0, -a], [-cos(α) * sin(θ), cos(α) * cos(θ), sin(α), -d
      * sin(α)], [sin(α) * sin(θ), -sin(α) * cos(θ), cos(α), -d * cos(α)], [0, 0, 0, 1]]);
```

$$H := \begin{bmatrix} \cos(\theta) & -\cos(\alpha)\sin(\theta) & \sin(\alpha)\sin(\theta) & a\cos(\theta) \\ \sin(\theta) & \cos(\alpha)\cos(\theta) & -\sin(\alpha)\cos(\theta) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Hinv := \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & -a \\ -\cos(\alpha)\sin(\theta) & \cos(\alpha)\cos(\theta) & \sin(\alpha) & -d\sin(\alpha) \\ \sin(\alpha)\sin(\theta) & -\sin(\alpha)\cos(\theta) & \cos(\alpha) & -d\cos(\alpha) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
**************************************************
Step 2: Construct the link frame transformation matrices...
**************************************************
Rule for D-H transformation:
1. translate along the z_(i-1) axis a distance d_i
2. rotate about z_(i-1) axis an angle theta_i to bring x_i axis into alignment with x_(i-1)  axis
3. translate along x_i axis a distance a_i
4. rotate about x_i axis an angle alpha_i

D-H Table:
----------------------------------------
link    alpha    a     d     theta
----------------------------------------
1       +Pi/2    a1    d1    theta 1
2       0        a2    d2    theta 2

> n := 2 :
> H01 := map(x→eval(subs(α =+ Pi/2, a = a1, d = d1, θ = θ1, x)), H);
  H12 := map(x→eval(subs(α = 0, a = a2, d = d2, θ = θ2, x)), H);
```

---

* Courtesy of Dr. Mark Karpenko, Guidance and Control Laboratory, Naval Post Graduate School, Monterey, CA.

$$H01 := \begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & a_1\cos(\theta_1) \\ \sin(\theta_1) & 0 & -\cos(\theta_1) & a_1\sin(\theta_1) \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H12 := \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2\sin(\theta_2) \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

****

Step 3: forward transformation matrices

****

```
> H02 := Multiply(H01, H12);
```

$$H02 := \left[\left[\cos(\theta_1)\cos(\theta_2), -\cos(\theta_1)\sin(\theta_2), \sin(\theta_1), \cos(\theta_1)\,a_2\cos(\theta_2) + \sin(\theta_1)\,d_2 + a_1\cos(\theta_1)\right], \left[\sin(\theta_1)\cos(\theta_2), -\sin(\theta_1)\sin(\theta_2), -\cos(\theta_1), \sin(\theta_1)\,a_2\cos(\theta_2) - \cos(\theta_1)\,d_2 + a_1\sin(\theta_1)\right], \left[\sin(\theta_2), \cos(\theta_2), 0, a_2\sin(\theta_2) + d_1\right], \left[0, 0, 0, 1\right]\right]$$

```
> q2x := H02[1, 4];
  q2y := H02[2, 4];
  q2z := H02[3, 4];
```

$$q2x := \cos(\theta_1)\,a_2\cos(\theta_2) + \sin(\theta_1)\,d_2 + a_1\cos(\theta_1)$$

$$q2y := \sin(\theta_1)\,a_2\cos(\theta_2) - \cos(\theta_1)\,d_2 + a_1\sin(\theta_1)$$

$$q2z := a_2\sin(\theta_2) + d_1$$

```
> CodeGeneration['Matlab'](q2x):
cg = cos(theta(1)) * a(2) * cos(theta(2)) + sin(theta(1)) * d
(2) + a(1) * cos(theta(1));
> CodeGeneration['Matlab'](q2y)
cg0 = sin(theta(1)) * a(2) * cos(theta(2)) - cos(theta(1)) * d
(2) + a(1) * sin(theta(1));
> CodeGeneration['Matlab'](q2z)
cg1 = a(2) * sin(theta(2)) + d(1);
```

****

Step 4: Link Jacobians

****

>

Rotation matrices

```
> R00 := IdentityMatrix(3);
  R01 := SubMatrix(H01, [1..3], [1..3]);
  R02 := SubMatrix(H02, [1..3], [1..3]);
```

$$R00 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R01 := \begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) \\ \sin(\theta_1) & 0 & -\cos(\theta_1) \\ 0 & 1 & 0 \end{bmatrix}$$

$$R02 := \begin{bmatrix} \cos(\theta_1)\cos(\theta_2) & -\cos(\theta_1)\sin(\theta_2) & \sin(\theta_1) \\ \sin(\theta_1)\cos(\theta_2) & -\sin(\theta_1)\sin(\theta_2) & -\cos(\theta_1) \\ \sin(\theta_2) & \cos(\theta_2) & 0 \end{bmatrix}$$

Position matrices

```
> r01 := SubMatrix(H01, [1..3], [4]);
  r12 := SubMatrix(H12, [1..3], [4]);
```

$$r01 := \begin{bmatrix} a_1\cos(\theta_1) \\ a_1\sin(\theta_1) \\ d_1 \end{bmatrix}$$

$$r12 := \begin{bmatrix} a_2\cos(\theta_2) \\ a_2\sin(\theta_2) \\ d_2 \end{bmatrix}$$

Z-vectors

```
> z_base := Vector([0, 0, 1]);
  z0 := Multiply(R00, z_base);
  z1 := Multiply(R01, z_base);
```

$$z_{base} := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$z0 := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$z1 := \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix}$$

Link mass-centers

```
> P11 := Matrix([[lx1], [ly1], [lz1]]);
  P22 := Matrix([[lx2], [ly2], [lz2]]);
```

$$P11 := \begin{bmatrix} lx_1 \\ ly_1 \\ lz_1 \end{bmatrix}$$

$$P22 := \begin{bmatrix} lx_2 \\ ly_2 \\ lz_2 \end{bmatrix}$$

```
>
```

Transformed mass-centers

```
> P11star := Multiply(R01, P11);
  P01star := Multiply(R00, r01) + P11star;
```

$$P11star := \begin{bmatrix} \cos(\theta_1)\, lx_1 + \sin(\theta_1)\, lz_1 \\ \sin(\theta_1)\, lx_1 - \cos(\theta_1)\, lz_1 \\ ly_1 \end{bmatrix}$$

$$P01star := \begin{bmatrix} a_1 \cos(\theta_1) + \cos(\theta_1)\, lx_1 + \sin(\theta_1)\, lz_1 \\ a_1 \sin(\theta_1) + \sin(\theta_1)\, lx_1 - \cos(\theta_1)\, lz_1 \\ d_1 + ly_1 \end{bmatrix}$$

```
> P22star := Multiply(R02, P22);
  P12star := combine( Multiply(R01, r12) ) + P22star;
  P02star := combine( Multiply(R00, r01) ) + P12star:
```

$$P22star := \begin{bmatrix} \cos(\theta_1)\cos(\theta_2)\,lx_2 - \cos(\theta_1)\sin(\theta_2)\,ly_2 + \sin(\theta_1)\,lz_2 \\ \sin(\theta_1)\cos(\theta_2)\,lx_2 - \sin(\theta_1)\sin(\theta_2)\,ly_2 - \cos(\theta_1)\,lz_2 \\ \sin(\theta_2)\,lx_2 + \cos(\theta_2)\,ly_2 \end{bmatrix}$$

$$P12star := \left[\left[\frac{1}{2}a_2\cos(\theta_1-\theta_2) + \frac{1}{2}a_2\cos(\theta_1+\theta_2) + \sin(\theta_1)\,d_2 + \cos(\theta_1)\cos(\theta_2)\,lx_2 - \cos(\theta_1)\sin(\theta_2)\,ly_2 + \sin(\theta_1)\,lz_2\right], \left[\frac{1}{2}a_2\sin(\theta_1+\theta_2) + \frac{1}{2}a_2\sin(\theta_1-\theta_2) - \cos(\theta_1)\,d_2 + \sin(\theta_1)\cos(\theta_2)\,lx_2 - \sin(\theta_1)\sin(\theta_2)\,ly_2 - \cos(\theta_1)\,lz_2\right], \left[a_2\sin(\theta_2) + \sin(\theta_2)\,lx_2 + \cos(\theta_2)\,ly_2\right]\right]$$

>

Build the Link Jacobians

```
> zeros := Transpose(Vector([0, 0, 0, 0, 0, 0])):
  JI11 := Transpose(Vector([z0 &x Vector([P01star]), z0])):
  JI21 := zeros:
  JI1 := combine((Transpose(Matrix([[JI11], [JI21]])))):
```

$$JI1 := \begin{bmatrix} -a_1\sin(\theta_1) - \sin(\theta_1)\,lx_1 + \cos(\theta_1)\,lz_1 & 0 \\ a_1\cos(\theta_1) + \cos(\theta_1)\,lx_1 + \sin(\theta_1)\,lz_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

```
> JI12 := Transpose(Vector([z0 &x Vector([P02star]), z0])):
  JI22 := Transpose(Vector([z1 &x Vector([P12star]), z1])):
  JI2 := combine(Transpose(Matrix([[JI12], [JI22]]))):
```

$$JI2 := \left[\left[-a_1\sin(\theta_1) - \frac{1}{2}a_2\sin(\theta_1+\theta_2) - \frac{1}{2}a_2\sin(\theta_1-\theta_2) + \cos(\theta_1)\,d_2 - \frac{1}{2}lx_2\sin(\theta_1+\theta_2) - \frac{1}{2}lx_2\sin(\theta_1-\theta_2) + \frac{1}{2}ly_2\cos(\theta_1-\theta_2) - \frac{1}{2}ly_2\cos(\theta_1+\theta_2) + \cos(\theta_1)\,lz_2, -\frac{1}{2}a_2\sin(\theta_1+\theta_2) + \frac{1}{2}a_2\sin(\theta_1-\theta_2) - \frac{1}{2}lx_2\sin(\theta_1+\theta_2) + \frac{1}{2}lx_2\sin(\theta_1-\theta_2) - \frac{1}{2}ly_2\cos(\theta_1-\theta_2) - \frac{1}{2}ly_2\cos(\theta_1+\theta_2)\right],$$

$$\left[a_1\cos(\theta_1)+\frac{1}{2}a_2\cos(\theta_1-\theta_2)+\frac{1}{2}a_2\cos(\theta_1+\theta_2)+\sin(\theta_1)d_2+\frac{1}{2}lx_2\cos(\theta_1\right.$$
$$-\theta_2)+\frac{1}{2}lx_2\cos(\theta_1+\theta_2)-\frac{1}{2}ly_2\sin(\theta_1+\theta_2)+\frac{1}{2}ly_2\sin(\theta_1-\theta_2)$$
$$+\sin(\theta_1)lz_2,-\frac{1}{2}a_2\cos(\theta_1-\theta_2)+\frac{1}{2}a_2\cos(\theta_1+\theta_2)-\frac{1}{2}lx_2\cos(\theta_1-\theta_2)$$
$$\left.+\frac{1}{2}lx_2\cos(\theta_1+\theta_2)-\frac{1}{2}ly_2\sin(\theta_1+\theta_2)-\frac{1}{2}ly_2\sin(\theta_1-\theta_2)\right],$$
$$\left[0,a_2\cos(\theta_2)+\cos(\theta_2)lx_2-\sin(\theta_2)ly_2\right],$$
$$\left[0,\sin(\theta_1)\right],$$
$$\left[0,-\cos(\theta_1)\right],$$
$$\left[1,0\right]\Big]$$

>

Link Inertia Matrices

```
> I11 := Matrix([[Ixx1, Ixy1, Ixz1], [Iyx1, Iyy1, Iyz1], [Izx1, Izy1, Izz1]]) :
  I1 := Multiply(R01, Multiply(I11, Transpose(R01)));
```

$$I1:=\Big[\Big[\cos(\theta_1)\left(Ixx1\cos(\theta_1)+Ixz1\sin(\theta_1)\right)+\sin(\theta_1)\left(Izx1\cos(\theta_1)+Izz1\sin(\theta_1)\right),$$
$$\cos(\theta_1)\left(Ixx1\sin(\theta_1)-Ixz1\cos(\theta_1)\right)+\sin(\theta_1)\left(Izx1\sin(\theta_1)-Izz1\cos(\theta_1)\right),$$
$$\cos(\theta_1)Ixy1+\sin(\theta_1)Izy1\Big],$$
$$\Big[\sin(\theta_1)\left(Ixx1\cos(\theta_1)+Ixz1\sin(\theta_1)\right)-\cos(\theta_1)\left(Izx1\cos(\theta_1)+Izz1\sin(\theta_1)\right),$$
$$\sin(\theta_1)\left(Ixx1\sin(\theta_1)-Ixz1\cos(\theta_1)\right)-\cos(\theta_1)\left(Izx1\sin(\theta_1)-Izz1\cos(\theta_1)\right),$$
$$\sin(\theta_1)Ixy1-\cos(\theta_1)Izy1\Big],$$
$$\Big[Iyx1\cos(\theta_1)+Iyz1\sin(\theta_1),Iyx1\sin(\theta_1)-Iyz1\cos(\theta_1),Iyy1\Big]\Big]$$

```
> I22 := Matrix([[Ixx2, Ixy2, Ixz2], [Iyx2, Iyy2, Iyz2], [Izx2, Izy2, Izz2]]) :
  I2 := Multiply(R02, Multiply(I22, Transpose(R02)));
```

$$I2:=\Big[\Big[\cos(\theta_1)\cos(\theta_2)\left(Ixx2\cos(\theta_1)\cos(\theta_2)-Ixy2\cos(\theta_1)\sin(\theta_2)+Ixz2\sin(\theta_1)\right)$$
$$-\cos(\theta_1)\sin(\theta_2)\left(Iyx2\cos(\theta_1)\cos(\theta_2)-Iyy2\cos(\theta_1)\sin(\theta_2)+Iyz2\sin(\theta_1)\right)$$
$$+\sin(\theta_1)\left(Izx2\cos(\theta_1)\cos(\theta_2)-Izy2\cos(\theta_1)\sin(\theta_2)+Izz2\sin(\theta_1)\right),$$
$$\cos(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_1)\cos(\theta_2)-Ixy2\sin(\theta_1)\sin(\theta_2)-Ixz2\cos(\theta_1)\right)$$
$$-\cos(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_1)\cos(\theta_2)-Iyy2\sin(\theta_1)\sin(\theta_2)-Iyz2\cos(\theta_1)\right)$$
$$+\sin(\theta_1)\left(Izx2\sin(\theta_1)\cos(\theta_2)-Izy2\sin(\theta_1)\sin(\theta_2)-Izz2\cos(\theta_1)\right),$$

$$\cos(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_2)+Ixy2\cos(\theta_2)\right)-\cos(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_2)+Iyy2\cos(\theta_2)\right)+\sin(\theta_1)\left(Izx2\sin(\theta_2)+Izy2\cos(\theta_2)\right)],$$

$$[\sin(\theta_1)\cos(\theta_2)\left(Ixx2\cos(\theta_1)\cos(\theta_2)-Ixy2\cos(\theta_1)\sin(\theta_2)+Ixz2\sin(\theta_1)\right)-\sin(\theta_1)\sin(\theta_2)\left(Iyx2\cos(\theta_1)\cos(\theta_2)-Iyy2\cos(\theta_1)\sin(\theta_2)+Iyz2\sin(\theta_1)\right)-\cos(\theta_1)\left(Izx2\cos(\theta_1)\cos(\theta_2)-Izy2\cos(\theta_1)\sin(\theta_2)+Izz2\sin(\theta_1)\right),$$

$$\sin(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_1)\cos(\theta_2)-Ixy2\sin(\theta_1)\sin(\theta_2)-Ixz2\cos(\theta_1)\right)-\sin(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_1)\cos(\theta_2)-Iyy2\sin(\theta_1)\sin(\theta_2)-Iyz2\cos(\theta_1)\right)-\cos(\theta_1)\left(Izx2\sin(\theta_1)\cos(\theta_2)-Izy2\sin(\theta_1)\sin(\theta_2)-Izz2\cos(\theta_1)\right),$$

$$\sin(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_2)+Ixy2\cos(\theta_2)\right)-\sin(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_2)+Iyy2\cos(\theta_2)\right)-\cos(\theta_1)\left(Izx2\sin(\theta_2)+Izy2\cos(\theta_2)\right)],$$

$$[\sin(\theta_2)\left(Ixx2\cos(\theta_1)\cos(\theta_2)-Ixy2\cos(\theta_1)\sin(\theta_2)+Ixz2\sin(\theta_1)\right)+\cos(\theta_2)\left(Iyx2\cos(\theta_1)\cos(\theta_2)-Iyy2\cos(\theta_1)\sin(\theta_2)+Iyz2\sin(\theta_1)\right),$$

$$\sin(\theta_2)\left(Ixx2\sin(\theta_1)\cos(\theta_2)-Ixy2\sin(\theta_1)\sin(\theta_2)-Ixz2\cos(\theta_1)\right)+\cos(\theta_2)\left(Iyx2\sin(\theta_1)\cos(\theta_2)-Iyy2\sin(\theta_1)\sin(\theta_2)-Iyz2\cos(\theta_1)\right),$$

$$\sin(\theta_2)\left(Ixx2\sin(\theta_2)+Ixy2\cos(\theta_2)\right)+\cos(\theta_2)\left(Iyx2\sin(\theta_2)+Iyy2\cos(\theta_2)\right)]]$$

```
>
```

exploit symmetry in inertia matrix...

```
> Iyx1 := Ixy1: Izx1 := Ixz1: Izy1 := Iyz1: I11:
> Iyx2 := Ixy2: Izx2 := Ixz2: Izy2 := Iyz2: I22:
```

Manipulator Inertia Matrix

Translational Energy

```
> Mt1 := Multiply(Transpose(SubMatrix(J11, [1..3], [1..n])), m1·SubMatrix(J11, [1..3], [1
      ..n]));
> Mt2 := Multiply(Transpose(SubMatrix(J12, [1..3], [1..n])), m2·SubMatrix(J12, [1..3], [1
      ..n]));
```

$$Mt2 := \left[\left[m_2\left(-a_1\sin(\theta_1)-\frac{1}{2}a_2\sin(\theta_1+\theta_2)-\frac{1}{2}a_2\sin(\theta_1-\theta_2)+\cos(\theta_1)d_2\right.\right.\right.$$

$$\left.-\frac{1}{2}lx_2\sin(\theta_1+\theta_2)-\frac{1}{2}lx_2\sin(\theta_1-\theta_2)+\frac{1}{2}ly_2\cos(\theta_1-\theta_2)-\frac{1}{2}ly_2\cos(\theta_1\right.$$

$$\left.+\theta_2)+\cos(\theta_1)lz_2\right)^2+m_2\left(a_1\cos(\theta_1)+\frac{1}{2}a_2\cos(\theta_1-\theta_2)+\frac{1}{2}a_2\cos(\theta_1\right.$$

$$\left.+\theta_2)+\sin(\theta_1)d_2+\frac{1}{2}lx_2\cos(\theta_1-\theta_2)+\frac{1}{2}lx_2\cos(\theta_1+\theta_2)-\frac{1}{2}ly_2\sin(\theta_1\right.$$

$$+\theta_2\Big)+\frac{1}{2}\,dy_2\sin\left(\theta_1-\theta_2\right)+\sin\left(\theta_1\right)dz_2\Big)^2,\left(-a_1\sin\left(\theta_1\right)-\frac{1}{2}\,a_2\sin\left(\theta_1+\theta_2\right)\right.$$
$$-\frac{1}{2}\,a_2\sin\left(\theta_1-\theta_2\right)+\cos\left(\theta_1\right)d_2-\frac{1}{2}\,dx_2\sin\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dx_2\sin\left(\theta_1-\theta_2\right)$$
$$+\frac{1}{2}\,dy_2\cos\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1+\theta_2\right)+\cos\left(\theta_1\right)dz_2\Big)\,m_2\left(-\frac{1}{2}\,a_2\sin\left(\theta_1+\theta_2\right)\right.$$
$$+\frac{1}{2}\,a_2\sin\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dx_2\sin\left(\theta_1+\theta_2\right)+\frac{1}{2}\,dx_2\sin\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1\right.$$
$$\left.-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1+\theta_2\right)\Big)+\left(a_1\cos\left(\theta_1\right)+\frac{1}{2}\,a_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,a_2\cos\left(\theta_1\right.\right.$$
$$+\theta_2\Big)+\sin\left(\theta_1\right)d_2+\frac{1}{2}\,dx_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,dx_2\cos\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dy_2\sin\left(\theta_1\right.$$
$$+\theta_2\Big)+\frac{1}{2}\,dy_2\sin\left(\theta_1-\theta_2\right)+\sin\left(\theta_1\right)dz_2\Big)\,m_2\left(-\frac{1}{2}\,a_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,a_2\cos\left(\theta_1\right.\right.$$
$$+\theta_2\Big)-\frac{1}{2}\,dx_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,dx_2\cos\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dy_2\sin\left(\theta_1+\theta_2\right)$$
$$\left.-\frac{1}{2}\,dy_2\sin\left(\theta_1-\theta_2\right)\Big)\right],$$
$$\left[\left(-a_1\sin\left(\theta_1\right)-\frac{1}{2}\,a_2\sin\left(\theta_1+\theta_2\right)-\frac{1}{2}\,a_2\sin\left(\theta_1-\theta_2\right)+\cos\left(\theta_1\right)d_2\right.$$
$$-\frac{1}{2}\,dx_2\sin\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dx_2\sin\left(\theta_1-\theta_2\right)+\frac{1}{2}\,dy_2\cos\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1\right.$$
$$+\theta_2\Big)+\cos\left(\theta_1\right)dz_2\Big)\,m_2\left(-\frac{1}{2}\,a_2\sin\left(\theta_1+\theta_2\right)+\frac{1}{2}\,a_2\sin\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dx_2\sin\left(\theta_1\right.\right.$$
$$+\theta_2\Big)+\frac{1}{2}\,dx_2\sin\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1+\theta_2\right)\Big)$$
$$+\left(a_1\cos\left(\theta_1\right)+\frac{1}{2}\,a_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,a_2\cos\left(\theta_1+\theta_2\right)+\sin\left(\theta_1\right)d_2\right.$$
$$+\frac{1}{2}\,dx_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,dx_2\cos\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dy_2\sin\left(\theta_1+\theta_2\right)+\frac{1}{2}\,dy_2\sin\left(\theta_1\right.$$
$$\left.-\theta_2\right)+\sin\left(\theta_1\right)dz_2\Big)\,m_2\left(-\frac{1}{2}\,a_2\cos\left(\theta_1-\theta_2\right)+\frac{1}{2}\,a_2\cos\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dx_2\cos\left(\theta_1\right.\right.$$
$$\left.-\theta_2\right)+\frac{1}{2}\,dx_2\cos\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dy_2\sin\left(\theta_1+\theta_2\right)-\frac{1}{2}\,dy_2\sin\left(\theta_1-\theta_2\right)\Big),\,m_2\left(\right.$$
$$-\frac{1}{2}\,a_2\sin\left(\theta_1+\theta_2\right)+\frac{1}{2}\,a_2\sin\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dx_2\sin\left(\theta_1+\theta_2\right)+\frac{1}{2}\,dx_2\sin\left(\theta_1\right.$$
$$\left.-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1-\theta_2\right)-\frac{1}{2}\,dy_2\cos\left(\theta_1+\theta_2\right)\Big)^2+m_2\left(-\frac{1}{2}\,a_2\cos\left(\theta_1-\theta_2\right)\right.$$

$$+\frac{1}{2}a_2\cos(\theta_1+\theta_2)-\frac{1}{2}lx_2\cos(\theta_1-\theta_2)+\frac{1}{2}lx_2\cos(\theta_1+\theta_2)-\frac{1}{2}ly_2\sin(\theta_1+\theta_2)-\frac{1}{2}ly_2\sin(\theta_1-\theta_2)\Big)^2+m_2\left(a_2\cos(\theta_2)+\cos(\theta_2)\,lx_2-\sin(\theta_2)\,ly_2\right)^2\Big]\Big]$$

>

Rotational Energy

> $Mr1 := Multiply(Transpose(SubMatrix(JI1, [4..6], [1..n])), Multiply(I1, SubMatrix(JI1, [4..6], [1..n])));$

$$Mr1 := \begin{bmatrix} Iyy1 & 0 \\ 0 & 0 \end{bmatrix}$$

> $Mr2 := Multiply(Transpose(SubMatrix(JI2, [4..6], [1..n])), Multiply(I2, SubMatrix(JI2, [4..6], [1..n])));$

$$Mr2 := \Big[\Big[\sin(\theta_2)\left(Ixx2\sin(\theta_2)+Ixy2\cos(\theta_2)\right)+\cos(\theta_2)\left(Iyx2\sin(\theta_2)+Iyy2\cos(\theta_2)\right),\left(\sin(\theta_2)\left(Ixx2\cos(\theta_1)\cos(\theta_2)-Ixy2\cos(\theta_1)\sin(\theta_2)+Ixz2\sin(\theta_1)\right)+\cos(\theta_2)\left(Iyx2\cos(\theta_1)\cos(\theta_2)-Iyy2\cos(\theta_1)\sin(\theta_2)+Iyz2\sin(\theta_1)\right)\right)\sin(\theta_1)-\left(\sin(\theta_2)\left(Ixx2\sin(\theta_1)\cos(\theta_2)-Ixy2\sin(\theta_1)\sin(\theta_2)-Ixz2\cos(\theta_1)\right)+\cos(\theta_2)\left(Iyx2\sin(\theta_1)\cos(\theta_2)-Iyy2\sin(\theta_1)\sin(\theta_2)-Iyz2\cos(\theta_1)\right)\right)\cos(\theta_1)\Big],$$

$$\Big[\sin(\theta_1)\left(\cos(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_2)+Ixy2\cos(\theta_2)\right)-\cos(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_2)+Iyy2\cos(\theta_2)\right)+\sin(\theta_1)\left(Izx2\sin(\theta_2)+Izy2\cos(\theta_2)\right)\right)-\cos(\theta_1)\left(\sin(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_2)+Ixy2\cos(\theta_2)\right)-\sin(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_2)+Iyy2\cos(\theta_2)\right)-\cos(\theta_1)\left(Izx2\sin(\theta_2)+Izy2\cos(\theta_2)\right)\right),\sin(\theta_1)\left(\left(\cos(\theta_1)\cos(\theta_2)\left(Ixx2\cos(\theta_1)\cos(\theta_2)-Ixy2\cos(\theta_1)\sin(\theta_2)+Ixz2\sin(\theta_1)\right)-\cos(\theta_1)\sin(\theta_2)\left(Iyx2\cos(\theta_1)\cos(\theta_2)-Iyy2\cos(\theta_1)\sin(\theta_2)+Iyz2\sin(\theta_1)\right)+\sin(\theta_1)\left(Izx2\cos(\theta_1)\cos(\theta_2)-Izy2\cos(\theta_1)\sin(\theta_2)+Izz2\sin(\theta_1)\right)\right)\sin(\theta_1)-\left(\cos(\theta_1)\cos(\theta_2)\left(Ixx2\sin(\theta_1)\cos(\theta_2)-Ixy2\sin(\theta_1)\sin(\theta_2)-Ixz2\cos(\theta_1)\right)-\cos(\theta_1)\sin(\theta_2)\left(Iyx2\sin(\theta_1)\cos(\theta_2)-Iyy2\sin(\theta_1)\sin(\theta_2)-Iyz2\cos(\theta_1)\right)+\sin(\theta_1)\left(Izx2\sin(\theta_1)\cos(\theta_2)-Izy2\sin(\theta_1)\sin(\theta_2)-Izz2\cos(\theta_1)\right)\right)\cos(\theta_1)\right)-\cos(\theta_1)\left(\left(\sin(\theta_1)\cos(\theta_2)\left(Ixx2\cos(\theta_1)\cos(\theta_2)-Ixy2\cos(\theta_1)\sin(\theta_2)+Ixz2\sin(\theta_1)\right)-\sin(\theta_1)\sin(\theta_2)\left(Iyx2\cos(\theta_1)\cos(\theta_2)-Iyy2\cos(\theta_1)\sin(\theta_2)\right)\right)$$

$$+ Iyz2\sin(\theta_1)) - \cos(\theta_1)\,(Izx2\cos(\theta_1)\cos(\theta_2) - Izy2\cos(\theta_1)\sin(\theta_2)$$
$$+ Izz2\sin(\theta_1))\sin(\theta_1) - (\sin(\theta_1)\cos(\theta_2)\,(Ixx2\sin(\theta_1)\cos(\theta_2)$$
$$- Ixy2\sin(\theta_1)\sin(\theta_2) - Ixz2\cos(\theta_1)) - \sin(\theta_1)\sin(\theta_2)\,(Iyx2\sin(\theta_1)\cos(\theta_2)$$
$$- Iyy2\sin(\theta_1)\sin(\theta_2) - Iyz2\cos(\theta_1)) - \cos(\theta_1)\,(Izx2\sin(\theta_1)\cos(\theta_2)$$
$$- Izy2\sin(\theta_1)\sin(\theta_2) - Izz2\cos(\theta_1)))\cos(\theta_1))]]$$

>

Total Energy

> $M := combine(Mt1 + Mt2 + Mr1 + Mr2)$:

Veclocity Coupling Vector

```
> V := Vector(n, 0) :
  for i from 1 to n do
    for j from 1 to n do
      for k from 1 to n do
        V_i := V_i + eval((diff(M_{i,j}, theta_k) - 1/2 * diff(M_{j,k}, theta_i)) * omega_j * omega_k);
      end do;
    end do;
  end do;
  combine(V);
```

$$\left[\left[\left(\frac{\partial}{\partial\theta_2}\left(Mt1 + \left[\left[2\,m_2\,d_2\,lz_2 - 2\,m_2\,a_1\,ly_2\sin(\theta_2) + \frac{1}{2}\,Ixx2 + 2\,m_2\,a_1\,a_2\cos(\theta_2)\right.\right.\right.\right.$$
$$+ 2\,m_2\,a_1\,lx_2\cos(\theta_2) - m_2\,lx_2\,ly_2\sin(2\,\theta_2) + m_2\,a_2\,lx_2 - \frac{1}{2}\,m_2\,ly_2^2\cos(2\,\theta_2) + \frac{1}{2}\,m_2$$
$$lx_2^2 + \frac{1}{2}\,m_2\,a_2^2 + \frac{1}{2}\,m_2\,ly_2^2 + \frac{1}{2}\,Iyx2\sin(2\,\theta_2) - m_2\,a_2\,ly_2\sin(2\,\theta_2)$$
$$+ m_2\,a_2\,lx_2\cos(2\,\theta_2) + \frac{1}{2}\,Iyy2\cos(2\,\theta_2) - \frac{1}{2}\,Ixx2\cos(2\,\theta_2) + Iyy1 + \frac{1}{2}\,Iyy2 + m_2$$
$$a_1^2 + m_2\,lz_2^2 + m_2\,d_2^2 + \frac{1}{2}\,m_2\,a_2^2\cos(2\,\theta_2) + \frac{1}{2}\,m_2\,lx_2^2\cos(2\,\theta_2) + \frac{1}{2}\,Ixy2\sin(2\,\theta_2),$$
$$-m_2\,d_2\,ly_2\cos(\theta_2) - m_2\,lz_2\,lx_2\sin(\theta_2) - m_2\,lz_2\,a_2\sin(\theta_2) - m_2\,lz_2\,ly_2\cos(\theta_2)$$
$$- m_2\,d_2\,a_2\sin(\theta_2) - m_2\,d_2\,lx_2\sin(\theta_2) + Iyz2\cos(\theta_2) + Ixz2\sin(\theta_2)\Big],$$

$$\left[Izx2\sin(\theta_2) + Izy2\cos(\theta_2) - m_2\,d_2\,ly_2\cos(\theta_2) - m_2\,lz_2\,lx_2\sin(\theta_2)\right.$$

$$- m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2) - m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2),$$

$$2\, m_2\, a_2\, lx_2 + m_2\, lx_2^2 + m_2\, a_2^2 + m_2\, ly_2^2 + Izz2\big]\big]\Big)\Big)\,\omega_1\,\omega_2 + \left(\frac{\partial}{\partial\theta_2}\left(Mf1\right.\right.$$

$$+ \Big[\Big[2\, m_2\, d_2\, lz_2 - 2\, m_2\, a_1\, ly_2 \sin(\theta_2) + \frac{1}{2}\, Ixx2^{1,1} + 2\, m_2\, a_1\, a_2 \cos(\theta_2)$$

$$+ 2\, m_2\, a_1\, lx_2 \cos(\theta_2) - m_2\, lx_2\, ly_2 \sin(2\,\theta_2) + m_2\, a_2\, lx_2 - \frac{1}{2}\, m_2\, ly_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2$$

$$lx_2^2 + \frac{1}{2}\, m_2\, a_2^2 + \frac{1}{2}\, m_2\, ly_2^2 + \frac{1}{2}\, Iyx2 \sin(2\,\theta_2) - m_2\, a_2\, ly_2 \sin(2\,\theta_2)$$

$$+ m_2\, a_2\, lx_2 \cos(2\,\theta_2) + \frac{1}{2}\, Iyy2 \cos(2\,\theta_2) - \frac{1}{2}\, Ixx2 \cos(2\,\theta_2) + Iyy1 + \frac{1}{2}\, Iyy2 + m_2$$

$$a_1^2 + m_2\, lz_2^2 + m_2\, d_2^2 + \frac{1}{2}\, m_2\, a_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2\, lx_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, Ixy2 \sin(2\,\theta_2),$$

$$- m_2\, d_2\, ly_2 \cos(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2) - m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2)$$

$$- m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2) + Iyz2 \cos(\theta_2) + Ixz2 \sin(\theta_2)\Big],$$

$$\Big[Izx2 \sin(\theta_2) + Izy2 \cos(\theta_2) - m_2\, d_2\, ly_2 \cos(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2)$$

$$- m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2) - m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2),$$

$$2\, m_2\, a_2\, lx_2 + m_2\, lx_2^2 + m_2\, a_2^2 + m_2\, ly_2^2 + Izz2\big]\big]\Big)\Big)\,\omega_2^2\Big],$$

$$\Big[-\frac{1}{2}\left(\frac{\partial}{\partial\theta_2}\left(Mf1 + \Big[\Big[2\, m_2\, d_2\, lz_2 - 2\, m_2\, a_1\, ly_2 \sin(\theta_2) + \frac{1}{2}\, Ixx2\right.\right.$$

$$+ 2\, m_2\, a_1\, a_2 \cos(\theta_2) + 2\, m_2\, a_1\, lx_2 \cos(\theta_2) - m_2\, lx_2\, ly_2 \sin(2\,\theta_2) + m_2\, a_2\, lx_2 - \frac{1}{2}\, m_2$$

$$ly_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2\, lx_2^2 + \frac{1}{2}\, m_2\, a_2^2 + \frac{1}{2}\, m_2\, ly_2^2 + \frac{1}{2}\, Iyx2 \sin(2\,\theta_2)$$

$$- m_2\, a_2\, ly_2 \sin(2\,\theta_2) + m_2\, a_2\, lx_2 \cos(2\,\theta_2) + \frac{1}{2}\, Iyy2 \cos(2\,\theta_2) - \frac{1}{2}\, Ixx2 \cos(2\,\theta_2)$$

$$+ Iyy1 + \frac{1}{2}\, Iyy2 + m_2\, a_1^2 + m_2\, lz_2^2 + m_2\, d_2^2 + \frac{1}{2}\, m_2\, a_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2$$

$$lx_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, Ixy2 \sin(2\,\theta_2),\ - m_2\, d_2\, ly_2 \cos(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2)$$

$$- m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2) - m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2)$$

$$+ Iyz2 \cos(\theta_2) + Ixz2 \sin(\theta_2)\Big],$$

$$\left[Izx2\sin\left(\theta_2\right) + Izy2\cos\left(\theta_2\right) - m_2\, d_2\, ly_2\cos\left(\theta_2\right) - m_2\, lz_2\, lx_2\sin\left(\theta_2\right)\right.$$

$$- m_2\, lz_2\, a_2\sin\left(\theta_2\right) - m_2\, lz_2\, ly_2\cos\left(\theta_2\right) - m_2\, d_2\, a_2\sin\left(\theta_2\right) - m_2\, d_2\, lx_2\sin\left(\theta_2\right),$$

$$\left.2\, m_2\, a_2\, lx_2 + m_2\, lx_2^2 + m_2\, a_2^2 + m_2\, ly_2^2 + Izz2\right]\Big]\Big)\quad\Big)\,\omega_1^2 + \left(\frac{\partial}{\partial\theta_2}\left(Mt1 + \left[\right.\right.$$

$$\left[2\, m_2\, d_2\, lz_2 - 2\, m_2\, a_1\, ly_2\sin\left(\theta_2\right) + \frac{1}{2}\, Ixx2 + \overset{1,\,1}{2}\, m_2\, a_1\, a_2\cos\left(\theta_2\right) + 2\, m_2\, a_1\, lx_2\cos\left(\theta_2\right)\right.$$

$$- m_2\, lx_2\, ly_2\sin\left(2\,\theta_2\right) + m_2\, a_2\, lx_2 - \frac{1}{2}\, m_2\, ly_2^2\cos\left(2\,\theta_2\right) + \frac{1}{2}\, m_2\, lx_2^2 + \frac{1}{2}\, m_2\, a_2^2$$

$$+ \frac{1}{2}\, m_2\, ly_2^2 + \frac{1}{2}\, Iyx2\sin\left(2\,\theta_2\right) - m_2\, a_2\, ly_2\sin\left(2\,\theta_2\right) + m_2\, a_2\, lx_2\cos\left(2\,\theta_2\right)$$

$$+ \frac{1}{2}\, Iyy2\cos\left(2\,\theta_2\right) - \frac{1}{2}\, Ixx2\cos\left(2\,\theta_2\right) + Iyy1 + \frac{1}{2}\, Iyy2 + m_2\, a_1^2 + m_2\, lz_2^2 + m_2\, d_2^2$$

$$+ \frac{1}{2}\, m_2\, a_2^2\cos\left(2\,\theta_2\right) + \frac{1}{2}\, m_2\, lx_2^2\cos\left(2\,\theta_2\right) + \frac{1}{2}\, Ixy2\sin\left(2\,\theta_2\right),\ -m_2\, d_2\, ly_2\cos\left(\theta_2\right)$$

$$- m_2\, lz_2\, lx_2\sin\left(\theta_2\right) - m_2\, lz_2\, a_2\sin\left(\theta_2\right) - m_2\, lz_2\, ly_2\cos\left(\theta_2\right) - m_2\, d_2\, a_2\sin\left(\theta_2\right)$$

$$\left.- m_2\, d_2\, lx_2\sin\left(\theta_2\right) + Iyz2\cos\left(\theta_2\right) + Ixz2\sin\left(\theta_2\right)\right],$$

$$\left[Izx2\sin\left(\theta_2\right) + Izy2\cos\left(\theta_2\right) - m_2\, d_2\, ly_2\cos\left(\theta_2\right) - m_2\, lz_2\, lx_2\sin\left(\theta_2\right)\right.$$

$$- m_2\, lz_2\, a_2\sin\left(\theta_2\right) - m_2\, lz_2\, ly_2\cos\left(\theta_2\right) - m_2\, d_2\, a_2\sin\left(\theta_2\right) - m_2\, d_2\, lx_2\sin\left(\theta_2\right),$$

$$\left.2\, m_2\, a_2\, lx_2 + m_2\, lx_2^2 + m_2\, a_2^2 + m_2\, ly_2^2 + Izz2\right]\Big]\Big)\quad - \frac{1}{2}\,\frac{\partial}{\partial\theta_2}\left(Mt1 + \left[\right.\right.$$

$$\left[2\, m_2\, d_2\, lz_2 - 2\, m_2\, a_1\, ly_2\sin\left(\theta_2\right) + \frac{1}{2}\, Ixx2 + \overset{2,\,1}{2}\, m_2\, a_1\, a_2\cos\left(\theta_2\right) + 2\, m_2\, a_1\, lx_2\cos\left(\theta_2\right)\right.$$

$$- m_2\, lx_2\, ly_2\sin\left(2\,\theta_2\right) + m_2\, a_2\, lx_2 - \frac{1}{2}\, m_2\, ly_2^2\cos\left(2\,\theta_2\right) + \frac{1}{2}\, m_2\, lx_2^2 + \frac{1}{2}\, m_2\, a_2^2$$

$$+ \frac{1}{2}\, m_2\, ly_2^2 + \frac{1}{2}\, Iyx2\sin\left(2\,\theta_2\right) - m_2\, a_2\, ly_2\sin\left(2\,\theta_2\right) + m_2\, a_2\, lx_2\cos\left(2\,\theta_2\right)$$

$$+\frac{1}{2}\,Iyy2\cos(2\,\theta_2)-\frac{1}{2}\,Ixx2\cos(2\,\theta_2)+Iyy1+\frac{1}{2}\,Iyy2+m_2\,a_1^2+m_2\,lz_2^2+m_2\,d_2^2$$
$$+\frac{1}{2}\,m_2\,a_2^2\cos(2\,\theta_2)+\frac{1}{2}\,m_2\,lx_2^2\cos(2\,\theta_2)+\frac{1}{2}\,Ixy2\sin(2\,\theta_2),\ -m_2\,d_2\,ly_2\cos(\theta_2)$$
$$-m_2\,lz_2\,lx_2\sin(\theta_2)-m_2\,lz_2\,a_2\sin(\theta_2)-m_2\,lz_2\,ly_2\cos(\theta_2)-m_2\,d_2\,a_2\sin(\theta_2)$$
$$-m_2\,d_2\,lx_2\sin(\theta_2)+Iyz2\cos(\theta_2)+Ixz2\sin(\theta_2)\,],$$

$$[\,Izx2\sin(\theta_2)+Izy2\cos(\theta_2)-m_2\,d_2\,ly_2\cos(\theta_2)-m_2\,lz_2\,lx_2\sin(\theta_2)$$

$$-m_2\,lz_2\,a_2\sin(\theta_2)-m_2\,lz_2\,ly_2\cos(\theta_2)-m_2\,d_2\,a_2\sin(\theta_2)-m_2\,d_2\,lx_2\sin(\theta_2),$$

$$2\,m_2\,a_2\,lx_2+m_2\,lx_2^2+m_2\,a_2^2+m_2\,ly_2^2+Izz2\,]\,]\,)\Big)_{1,\,2}\,\omega_1\,\omega_2-\frac{1}{2}\left(\frac{\partial}{\partial\theta_2}\left(Mt1\right.\right.$$
$$+\,[\,[\,[\,2\,m_2\,d_2\,lz_2-2\,m_2\,a_1\,ly_2\sin(\theta_2)+\frac{1}{2}\,Ixx2+2\,m_2\,a_1\,a_2\cos(\theta_2)$$
$$+2\,m_2\,a_1\,lx_2\cos(\theta_2)-m_2\,lx_2\,ly_2\sin(2\,\theta_2)+m_2\,a_2\,lx_2-\frac{1}{2}\,m_2\,ly_2^2\cos(2\,\theta_2)+\frac{1}{2}\,m_2$$
$$lx_2^2+\frac{1}{2}\,m_2\,a_2^2+\frac{1}{2}\,m_2\,ly_2^2+\frac{1}{2}\,Iyx2\sin(2\,\theta_2)-m_2\,a_2\,ly_2\sin(2\,\theta_2)$$
$$+m_2\,a_2\,lx_2\cos(2\,\theta_2)+\frac{1}{2}\,Iyy2\cos(2\,\theta_2)-\frac{1}{2}\,Ixx2\cos(2\,\theta_2)+Iyy1+\frac{1}{2}\,Iyy2+m_2$$
$$a_1^2+m_2\,lz_2^2+m_2\,d_2^2+\frac{1}{2}\,m_2\,a_2^2\cos(2\,\theta_2)+\frac{1}{2}\,m_2\,lx_2^2\cos(2\,\theta_2)+\frac{1}{2}\,Ixy2\sin(2\,\theta_2),$$
$$-m_2\,d_2\,ly_2\cos(\theta_2)-m_2\,lz_2\,lx_2\sin(\theta_2)-m_2\,lz_2\,a_2\sin(\theta_2)-m_2\,lz_2\,ly_2\cos(\theta_2)$$
$$-m_2\,d_2\,a_2\sin(\theta_2)-m_2\,d_2\,lx_2\sin(\theta_2)+Iyz2\cos(\theta_2)+Ixz2\sin(\theta_2)\,],$$

$$[\,Izx2\sin(\theta_2)+Izy2\cos(\theta_2)-m_2\,d_2\,ly_2\cos(\theta_2)-m_2\,lz_2\,lx_2\sin(\theta_2)$$

$$- m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2) - m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2),$$

$$2\, m_2\, a_2\, lx_2 + m_2\, lx_2^2 + m_2\, a_2^2 + m_2\, ly_2^2 + Izz2]])\quad)\,\omega_2\,\omega_1 + \frac{1}{2}\left(\frac{\partial}{\partial\theta_2}\left(Mt1\right.\right.$$

$$_{2,1}$$

$$+ [[2\, m_2\, d_2\, lz_2 - 2\, m_2\, a_1\, ly_2 \sin(\theta_2) + \frac{1}{2}\, Ixx2 + 2\, m_2\, a_1\, a_2 \cos(\theta_2)$$

$$+ 2\, m_2\, a_1\, lx_2 \cos(\theta_2) - m_2\, lx_2\, ly_2 \sin(2\,\theta_2) + m_2\, a_2\, lx_2 - \frac{1}{2}\, m_2\, ly_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2$$

$$lx_2^2 + \frac{1}{2}\, m_2\, a_2^2 + \frac{1}{2}\, m_2\, ly_2^2 + \frac{1}{2}\, Iyx2 \sin(2\,\theta_2) - m_2\, a_2\, ly_2 \sin(2\,\theta_2)$$

$$+ m_2\, a_2\, lx_2 \cos(2\,\theta_2) + \frac{1}{2}\, Iyy2 \cos(2\,\theta_2) - \frac{1}{2}\, Ixx2 \cos(2\,\theta_2) + Iyy1 + \frac{1}{2}\, Iyy2 + m_2$$

$$a_1^2 + m_2\, lz_2^2 + m_2\, d_2^2 + \frac{1}{2}\, m_2\, a_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2\, lx_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, Ixy2 \sin(2\,\theta_2),$$

$$-m_2\, d_2\, ly_2 \cos(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2) - m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2)$$

$$- m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2) + Iyz2 \cos(\theta_2) + Ixz2 \sin(\theta_2)],$$

$$[Izx2 \sin(\theta_2) + Izy2 \cos(\theta_2) - m_2\, d_2\, ly_2 \cos(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2)$$

$$- m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2) - m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2),$$

$$2\, m_2\, a_2\, lx_2 + m_2\, lx_2^2 + m_2\, a_2^2 + m_2\, ly_2^2 + Izz2]])\quad)\,\omega_2^2]]$$

$$_{2,2}$$

Gravity Vector

```
> G := Vector(n, 0) :
  g_vec := Vector([0, 0, -gc]) :
  i := 1 :
  for j from 1 to n do
    G_j := G_j - m_i * Multiply(Transpose(g_vec), Vector([SubMatrix(JI1, [1..3], [j])])) :
  end do:
  i := 2 :
  for j from 1 to n do
    G_j := G_j - m_i * Multiply(Transpose(g_vec), Vector([SubMatrix(JI2, [1..3], [j])])) :
  end do:
  G;
```

$$\begin{bmatrix} 0 \\ m_2\,(a_2 \cos(\theta_2) + \cos(\theta_2)\, lx_2 - \sin(\theta_2)\, ly_2)\, gc \end{bmatrix}$$

Assemble the dynamic equations

```
>
> T := Vector(n, 0) :
  T := Multiply(M, Vector([α1, α2])) + G + V:
  T := combine(T) :
  T := collect(T, {α1, α2, gc}) :
>
> T1 := T1;
```

$$T_1 := \left(-m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2)\right.$$
$$- m_2\, d_2\, ly_2 \cos(\theta_2) + Ixz2 \sin(\theta_2) + Iyz2 \cos(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2)\Big)\, \alpha_2 + \Big(Iyy1$$
$$+ 2\, m_2\, d_2\, lz_2 + Ixy2 \sin(2\,\theta_2) + m_2\, a_1^2 + m_2\, lz_2^2 + \frac{1}{2}\, Iyy2 + 2\, m_1\, a_1\, lx_1$$
$$+ \frac{1}{2}\, Iyy2 \cos(2\,\theta_2) - \frac{1}{2}\, m_2\, ly_2^2 \cos(2\,\theta_2) + \frac{1}{2}\, m_2\, a_2^2 \cos(2\,\theta_2) + m_2\, a_2\, lx_2 \cos(2\,\theta_2)$$
$$- m_2\, a_2\, ly_2 \sin(2\,\theta_2) + m_2\, d_2^2 + \frac{1}{2}\, m_2\, lx_2^2 \cos(2\,\theta_2) + 2\, m_2\, a_1\, lx_2 \cos(\theta_2)$$
$$- m_2\, lx_2\, ly_2 \sin(2\,\theta_2) + m_1\, a_1^2 + m_1\, lx_1^2 + m_1\, lz_1^2 - 2\, m_2\, a_1\, ly_2 \sin(\theta_2)$$
$$- \frac{1}{2}\, Ixx2 \cos(2\,\theta_2) + 2\, m_2\, a_1\, a_2 \cos(\theta_2) + \frac{1}{2}\, Ixx2 + m_2\, a_2\, lx_2 + \frac{1}{2}\, m_2\, a_2^2 + \frac{1}{2}\, m_2$$
$$lx_2^2 + \frac{1}{2}\, m_2\, ly_2^2\Big)\, \alpha_1 - \omega_2^2\, Iyz2 \sin(\theta_2) - \omega_1\, \omega_2\, m_2\, lx_2^2 \sin(2\,\theta_2) + \omega_2^2\, Ixz2 \cos(\theta_2)$$
$$+ \omega_1\, \omega_2\, m_2\, ly_2^2 \sin(2\,\theta_2) + 2\, \omega_1\, \omega_2\, Ixy2 \cos(2\,\theta_2) - 2\, \omega_1\, \omega_2\, m_2\, lx_2\, ly_2 \cos(2\,\theta_2) +$$
$$\omega_2^2\, m_2\, d_2\, ly_2 \sin(\theta_2) - \omega_1\, \omega_2\, Iyy2 \sin(2\,\theta_2) + \omega_1\, \omega_2\, Ixx2 \sin(2\,\theta_2) +$$
$$\omega_2^2\, m_2\, lz_2\, ly_2 \sin(\theta_2) - \omega_2^2\, m_2\, d_2\, lx_2 \cos(\theta_2) - \omega_1\, \omega_2\, m_2\, a_2^2 \sin(2\,\theta_2)$$
$$- 2\, \omega_1\, \omega_2\, m_2\, a_1\, lx_2 \sin(\theta_2) - \omega_2^2\, m_2\, lz_2\, lx_2 \cos(\theta_2) - 2\, \omega_1\, \omega_2\, m_2\, a_2\, ly_2 \cos(2\,\theta_2) -$$
$$\omega_2^2\, m_2\, d_2\, a_2 \cos(\theta_2) - 2\, \omega_1\, \omega_2\, m_2\, a_2\, lx_2 \sin(2\,\theta_2) - 2\, \omega_1\, \omega_2\, m_2\, a_1\, a_2 \sin(\theta_2) -$$
$$\omega_2^2\, m_2\, lz_2\, a_2 \cos(\theta_2) - 2\, \omega_1\, \omega_2\, m_2\, a_1\, ly_2 \cos(\theta_2)$$

```
> CodeGeneration['Matlab'](T1);
cg2 = (-m(2) * d(2) * a(2) * sin(theta(2)) - m(2) * lz(2) * a
(2) * sin(theta(2)) - m(2) * lz(2) * lx(2) * sin(theta(2)) - m
(2) * lz(2) * ly(2) * cos(theta(2)) - m(2) * d(2) * ly(2) * cos
(theta(2)) + Ixz2 * sin(theta(2)) + Iyz2 * cos(theta(2)) - m(2)
* d(2) * lx(2) * sin(theta(2))) * alpha(2) + (Iyy1 + 0.2e1 * m
(2) * d(2) * lz(2) + Ixy2 * sin(0.2e1 * theta(2)) + m(2) * a(1)
^ 2 + m(2) * lz(2) ^ 2 + Iyy2 / 0.2e1 + 0.2e1 * m(1) * a(1) *
lx(1) + Iyy2 * cos(0.2e1 * theta(2)) / 0.2e1 - m(2) * ly(2) ^ 2
* cos(0.2e1 * theta(2)) / 0.2e1 + m(2) * a(2) ^ 2 * cos(0.2e1 *
theta(2)) / 0.2e1 + m(2) * a(2) * lx(2) * cos(0.2e1 * theta(2))
```

```
- m(2) * a(2) * ly(2) * sin(0.2e1 * theta(2)) + m(2) * d(2) ^ 2
+ m(2) * lx(2) ^ 2 * cos(0.2e1 * theta(2)) / 0.2e1 + 0.2e1 * m
(2) * a(1) * lx(2) * cos(theta(2)) - m(2) * lx(2) * ly(2) * sin
(0.2e1 * theta(2)) + m(1) * a(1) ^ 2 + m(1) * lx(1) ^ 2 + m(1)
* lz(1) ^ 2 - 0.2e1 * m(2) * a(1) * ly(2) * sin(theta(2)) -
Ixx2 * cos(0.2e1 * theta(2)) / 0.2e1 + 0.2e1 * m(2) * a(1) * a
(2) * cos(theta(2)) + Ixx2 / 0.2e1 + m(2) * a(2) * lx(2) + m(2)
* a(2) ^ 2 / 0.2e1 + m(2) * lx(2) ^ 2 / 0.2e1 + m(2) * ly(2) ^
2 / 0.2e1) * alpha(1) - omega(2) ^ 2 * Iyz2 * sin(theta(2)) -
omega(1) * omega(2) * m(2) * lx(2) ^ 2 * sin(0.2e1 * theta(2))
+ omega(2) ^ 2 * Ixz2 * cos(theta(2)) + omega(1) * omega(2) * m
(2) * ly(2) ^ 2 * sin(0.2e1 * theta(2)) + 0.2e1 * omega(1) *
omega(2) * Ixy2 * cos(0.2e1 * theta(2)) - 0.2e1 * omega(1) *
omega(2) * m(2) * lx(2) * ly(2) * cos(0.2e1 * theta(2)) + omega
(2) ^ 2 * m(2) * d(2) * ly(2) * sin(theta(2)) - omega(1) *
omega(2) * Iyy2 * sin(0.2e1 * theta(2)) + omega(1) * omega(2) *
Ixx2 * sin(0.2e1 * theta(2)) + omega(2) ^ 2 * m(2) * lz(2) * ly
(2) * sin(theta(2)) - omega(2) ^ 2 * m(2) * d(2) * lx(2) * cos
(theta(2)) - omega(1) * omega(2) * m(2) * a(2) ^ 2 * sin(0.2e1
* theta(2)) - 0.2e1 * omega(1) * omega(2) * m(2) * a(1) * lx(2)
* sin(theta(2)) - omega(2) ^ 2 * m(2) * lz(2) * lx(2) * cos
(theta(2)) - 0.2e1 * omega(1) * omega(2) * m(2) * a(2) * ly(2)
* cos(0.2e1 * theta(2)) - omega(2) ^ 2 * m(2) * d(2) * a(2) *
cos(theta(2)) - 0.2e1 * omega(1) * omega(2) * m(2) * a(2) * lx
(2) * sin(0.2e1 * theta(2)) - 0.2e1 * omega(1) * omega(2) * m
(2) * a(1) * a(2) * sin(theta(2)) - omega(2) ^ 2 * m(2) * lz(2)
* a(2) * cos(theta(2)) - 0.2e1 * omega(1) * omega(2) * m(2) * a
(1) * ly(2) * cos(theta(2));
```

```
> T2 := T2;
```

$$T_2 := \left(-m_2\, ly_2 \sin(\theta_2) + m_2\, a_2 \cos(\theta_2) + m_2\, lx_2 \cos(\theta_2)\right) gc + \left(Izz2 + 2\, m_2\, a_2\, lx_2 + m_2\, a_2^2 + m_2\, lx_2^2 + m_2\, ly_2^2\right) \alpha_2 + \left(-m_2\, d_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, a_2 \sin(\theta_2) - m_2\, lz_2\, lx_2 \sin(\theta_2) - m_2\, lz_2\, ly_2 \cos(\theta_2) - m_2\, d_2\, ly_2 \cos(\theta_2) + Ixz2 \sin(\theta_2) + Iyz2 \cos(\theta_2) - m_2\, d_2\, lx_2 \sin(\theta_2)\right) \alpha_1 + \omega_1^2\, m_2\, a_2\, ly_2 \cos(2\,\theta_2) - \frac{1}{2}\,\omega_1^2\, Ixx2 \sin(2\,\theta_2) + \frac{1}{2}\,\omega_1^2\, m_2\, lx_2^2 \sin(2\,\theta_2) + \omega_1^2\, m_2\, a_2\, lx_2 \sin(2\,\theta_2) + \omega_1^2\, m_2\, a_1\, a_2 \sin(\theta_2) + \omega_1^2\, m_2\, a_1\, lx_2 \sin(\theta_2) + \omega_1^2\, m_2\, lx_2\, ly_2 \cos(2\,\theta_2) + \omega_1^2\, m_2\, a_1\, ly_2 \cos(\theta_2) - \omega_1^2\, Ixy2 \cos(2\,\theta_2) + \frac{1}{2}\,\omega_1^2\, Iyy2 \sin(2\,\theta_2) - \frac{1}{2}\,\omega_1^2\, m_2\, ly_2^2 \sin(2\,\theta_2) + \frac{1}{2}\,\omega_1^2\, m_2\, a_2^2 \sin(2\,\theta_2)$$

```
> CodeGeneration['Matlab'](T2);
```

```
cg3 = (-m(2) * ly(2) * sin(theta(2)) + m(2) * a(2) * cos(theta
(2)) + m(2) * lx(2) * cos(theta(2))) * gc + (Izz2 + 0.2e1 * m
(2) * a(2) * lx(2) + m(2) * a(2) ^ 2 + m(2) * lx(2) ^ 2 + m(2)
* ly(2) ^ 2) * alpha(2) + (-m(2) * d(2) * a(2) * sin(theta(2))
- m(2) * lz(2) * a(2) * sin(theta(2)) - m(2) * lz(2) * lx(2) *
sin(theta(2)) - m(2) * lz(2) * ly(2) * cos(theta(2)) - m(2) * d
(2) * ly(2) * cos(theta(2)) + Ixz2 * sin(theta(2)) + Iyz2 * cos
(theta(2)) - m(2) * d(2) * lx(2) * sin(theta(2))) * alpha(1) +
omega(1) ^ 2 * m(2) * a(2) * ly(2) * cos(0.2e1 * theta(2)) -
omega(1) ^ 2 * Ixx2 * sin(0.2e1 * theta(2)) / 0.2e1 + omega(1)
^ 2 * m(2) * lx(2) ^ 2 * sin(0.2e1 * theta(2)) / 0.2e1 + omega
```

```
(1) ^ 2 * m(2) * a(2) * lx(2) * sin(0.2e1 * theta(2)) + omega
(1) ^ 2 * m(2) * a(1) * a(2) * sin(theta(2)) + omega(1) ^ 2 * m
(2) * a(1) * lx(2) * sin(theta(2)) + omega(1) ^ 2 * m(2) * lx
(2) * ly(2) * cos(0.2e1 * theta(2)) + omega(1) ^ 2 * m(2) * a
(1) * ly(2) * cos(theta(2)) - omega(1) ^ 2 * Ixy2 * cos(0.2e1 *
theta(2)) + omega(1) ^ 2 * Iyy2 * sin(0.2e1 * theta(2)) / 0.2e1
- omega(1) ^ 2 * m(2) * ly(2) ^ 2 * sin(0.2e1 * theta(2)) /
0.2e1 + omega(1) ^ 2 * m(2) * a(2) ^ 2 * sin(0.2e1 * theta(2))
/ 0.2e1;
>
>
>
> CodeGeneration['Matlab'](M[1, 1]);
cg4 = Iyy1 + (2 * m(2) * d(2) * lz(2)) + Ixy2 * sin((2 * theta
(2))) + (m(2) * a(1) ^ 2) + (m(2) * lz(2) ^ 2) + Iyy2 / 0.2e1 +
(2 * m(1) * a(1) * lx(1)) + Iyy2 * cos((2 * theta(2))) / 0.2e1
- m(2) * ly(2) ^ 2 * cos((2 * theta(2))) / 0.2e1 + m(2) * (a(2)
^ 2) * cos((2 * theta(2))) / 0.2e1 + m(2) * a(2) * lx(2) * cos(
(2 * theta(2))) - m(2) * a(2) * ly(2) * sin((2 * theta(2))) +
(m(2) * d(2) ^ 2) + m(2) * (lx(2) ^ 2) * cos((2 * theta(2))) /
0.2e1 + 0.2e1 * m(2) * a(1) * lx(2) * cos(theta(2)) - m(2) * lx
(2) * ly(2) * sin((2 * theta(2))) + (m(1) * a(1) ^ 2) + (m(1) *
lx(1) ^ 2) + (m(1) * lz(1) ^ 2) - 0.2e1 * m(2) * a(1) * ly(2) *
sin(theta(2)) - Ixx2 * cos((2 * theta(2))) / 0.2e1 + 0.2e1 * m
(2) * a(1) * a(2) * cos(theta(2)) + Ixx2 / 0.2e1 + (m(2) * a(2)
* lx(2)) + (m(2) * a(2) ^ 2) / 0.2e1 + (m(2) * lx(2) ^ 2) /
0.2e1 + m(2) * ly(2) ^ 2 / 0.2e1;
> CodeGeneration['Matlab'](M[1, 2]);
cg5 = -m(2) * d(2) * a(2) * sin(theta(2)) - m(2) * lz(2) * a(2)
* sin(theta(2)) - m(2) * lz(2) * lx(2) * sin(theta(2)) - m(2) *
lz(2) * ly(2) * cos(theta(2)) - m(2) * d(2) * ly(2) * cos(theta
(2)) + Ixz2 * sin(theta(2)) + Iyz2 * cos(theta(2)) - m(2) * d
(2) * lx(2) * sin(theta(2));
>
> CodeGeneration['Matlab'](M[2, 1]);
cg6 = -m(2) * d(2) * a(2) * sin(theta(2)) - m(2) * lz(2) * a(2)
* sin(theta(2)) - m(2) * lz(2) * lx(2) * sin(theta(2)) - m(2) *
lz(2) * ly(2) * cos(theta(2)) - m(2) * d(2) * ly(2) * cos(theta
(2)) + Ixz2 * sin(theta(2)) + Iyz2 * cos(theta(2)) - m(2) * d
(2) * lx(2) * sin(theta(2));
> CodeGeneration['Matlab'](M[2, 2]);
cg7 = Izz2 + 2 * m(2) * a(2) * lx(2) + m(2) * a(2) ^ 2 + m(2) *
lx(2) ^ 2 + m(2) * ly(2) ^ 2;
>
> CodeGeneration['Matlab'](G[1]);
cg8 = 0;
> CodeGeneration['Matlab'](G[2]);
cg9 = m(2) * (a(2) * cos(theta(2)) + lx(2) * cos(theta(2)) - ly
(2) * sin(theta(2))) * gc;
>
> CodeGeneration['Matlab'](V[1]);
cg10 = (-0.2e1 * m(2) * a(1) * lx(2) * sin(theta(2)) + 0.2e1 *
Ixy2 * cos(0.2e1 * theta(2)) - Iyy2 * sin(0.2e1 * theta(2)) -
```

```
0.2e1 * m(2) * a(1) * ly(2) * cos(theta(2)) - 0.2e1 * m(2) * lx
(2) * ly(2) * cos(0.2e1 * theta(2)) + Ixx2 * sin(0.2e1 * theta
(2)) - 0.2e1 * m(2) * a(1) * a(2) * sin(theta(2)) - m(2) * a(2)
^ 2 * sin(0.2e1 * theta(2)) - m(2) * lx(2) ^ 2 * sin(0.2e1 *
theta(2)) + m(2) * ly(2) ^ 2 * sin(0.2e1 * theta(2)) - 0.2e1 *
m(2) * a(2) * lx(2) * sin(0.2e1 * theta(2)) - 0.2e1 * m(2) * a
(2) * ly(2) * cos(0.2e1 * theta(2))) * omega(1) * omega(2) + (m
(2) * lz(2) * ly(2) * sin(theta(2)) + m(2) * d(2) * ly(2) * sin
(theta(2)) - m(2) * lz(2) * lx(2) * cos(theta(2)) - m(2) * lz
(2) * a(2) * cos(theta(2)) - m(2) * d(2) * lx(2) * cos(theta(2)
) + Ixz2 * cos(theta(2)) - Iyz2 * sin(theta(2)) - m(2) * d(2) *
a(2) * cos(theta(2))) * omega(2) ^ 2;
```

```
> CodeGeneration['Matlab'](V2);
```

```
cg11 = (m(2) * a(1) * lx(2) * sin(theta(2)) - Ixy2 * cos(0.2e1
* theta(2)) + Iyy2 * sin(0.2e1 * theta(2)) / 0.2e1 + m(2) * a
(1) * ly(2) * cos(theta(2)) + m(2) * lx(2) * ly(2) * cos(0.2e1
* theta(2)) - Ixx2 * sin(0.2e1 * theta(2)) / 0.2e1 + m(2) * a
(1) * a(2) * sin(theta(2)) + m(2) * a(2) ^ 2 * sin(0.2e1 *
theta(2)) / 0.2e1 + m(2) * lx(2) ^ 2 * sin(0.2e1 * theta(2)) /
0.2e1 - m(2) * ly(2) ^ 2 * sin(0.2e1 * theta(2)) / 0.2e1 + m(2)
* a(2) * lx(2) * sin(0.2e1 * theta(2)) + m(2) * a(2) * ly(2) *
cos(0.2e1 * theta(2))) * omega(1) ^ 2 + (m(2) * lz(2) * ly(2) *
sin(theta(2)) / 0.2e1 + m(2) * d(2) * ly(2) * sin(theta(2)) /
0.2e1 - m(2) * lz(2) * lx(2) * cos(theta(2)) / 0.2e1 - m(2) *
lz(2) * a(2) * cos(theta(2)) / 0.2e1 - m(2) * d(2) * lx(2) *
cos(theta(2)) / 0.2e1 + Ixz2 * cos(theta(2)) / 0.2e1 - Iyz2 *
sin(theta(2)) / 0.2e1 - m(2) * d(2) * a(2) * cos(theta(2)) /
0.2e1) * omega(1) * omega(2) + (-m(2) * lz(2) * ly(2) * sin
(theta(2)) / 0.2e1 - m(2) * d(2) * ly(2) * sin(theta(2)) /
0.2e1 + m(2) * lz(2) * lx(2) * cos(theta(2)) / 0.2e1 + m(2) *
lz(2) * a(2) * cos(theta(2)) / 0.2e1 + m(2) * d(2) * lx(2) *
cos(theta(2)) / 0.2e1 - Ixz2 * cos(theta(2)) / 0.2e1 + Iyz2 *
sin(theta(2)) / 0.2e1 + m(2) * d(2) * a(2) * cos(theta(2)) /
0.2e1) * omega(2) * omega(1);
```

```
>
```

LIST OF REFERENCES

[1] D. W. Johnson, and E. G. Gilbert, “Minimum Time Robot Path Planning in the Presence of Obstacles,” *Proceedings of the 24th Conference on Decision and Control*. Ft. Lauderdale, FL, December 1985, pp. 1748-1753.

[2] O. von Stryk and M. Schlemmerm “Optimal control of the industrial robot Manutec R3,” *International Series of Numerical Mathematics*, 115 (Basel: Birhauser, 1994), pp. 367-382.

[3] M. C. Steinbach, H. G. Bock, G. V. Kostin, and R. W. Longman, “Mathematical optimization in robotics: Towards automated high speed motion planning,” *Surveys on Mathematics for Industry*, vol. 7 no. 4, pp. 303-340, 1998.

[4] NASA, “Space Shuttle Photo Gallery,” http://www.nasa.gov/mission_pages/shuttle/behindscenes/rms_gallery.html.

[5] A.B. Bosse et al., “SUMO: spacecraft for the universal modification of orbits.” *Proceedings of SPIE,* vol. SPIE-5419, pp. 36-46, 2004. http://projects.nrl.navy.mil/sumo/whitePapers/SUMOPaperSPIE5419-7_29Mar04_ver2.doc.

[6] B. K. Kim and K. G. Shin, “Minimum-time path planning for robot arms and their dynamics,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 213-223, March/April 1985.

[7] H. Choset et al., *Principles of Robot Motion: Theory, Algorithms, and Implementaion*. Cambridge: The Massachusetts Institute of Technology Press, 2005.

[8] T. Chettibi, H. E. Lehtihet, M. Haddadm and S. Hanchi, “Minimum cost trajectory planning for industrial robots,” *European Journal of Mechanics and Solids*, vol. 23, pp. 703-715, 2004.

[9] I. M. Ross, Control and Optimization: An introduction to Principles and Applications, Electronic Edition, Naval Postgraduate School, Monterey, CA, December, 2005.

[10] K. G. Shin and N. D. McKay, “Minimum-time control of robotic manipulators with geometric path constraints,” *IEEE Transactions on Automatic Control*, vol. AC-30, no. 6., pp. 531-541, June 1985.

[11] B. J. Martin and J. E. Bobrow, “Minimum-effort motions for open-chain manipulators with task-dependent end-effector constraints,” *The International Journal of Robotics Research*, vol. 18 no. 2, pp. 213-224 , February 1999.

[12] S. F. P. Saramago and V.S. Junior, “Optimal trajectory planning of robot manipulators in the presence of moving obstacles,” *Mechanism and Machine Theory*, vol. 35, pp. 1079-1094, 2000.

[13] T. Chettibi, “Synthesis of dynamic motions for robotic manipulators with geometric path constraints,” *Mechatronics*, vol. 16, pp. 547-563, 2006

[14] B. Faverjon and P. Tournassoud, “A local based approach for path planning of manipulators with a high number of degrees of freedom,” *1987 IEEE International Conference on Robotics and Automation. Proceedings*, Vol. 4, March 1987, pp. 1152-1159.

[15] J. E. Bobrow et al. “Optimal robot motions for physical criteria,” *Journal of Robotic Systems*, vol. 18, no. 12, pp. 785-795, 2001.

[16] B. Faverjon and P. Tournassoud, “A local based approach for path planning of manipulators with a high number of degrees of freedom,” *1987 IEEE International Conference on Robotics and Automation. Proceedings*, Vol. 4, March 1987, pp. 1152-1159.

[17] Z. Shiller, and H. H. Lu, “Computation of path constrained time optimal motions of robotic manipulators in the presence of obstacles,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 785-797, December, 1991.

[18] L. Pontryagin, V. Boltayansky, R. Gamkrelitze, and E. Mishchenko, *The Mathematical Theory of Optimal Processes*, Wiley, New York, 1962.

[19] R. Callies and P. Rentrop, “Optimal Control of Rigid-Link Manipulators by Indirect-Methods,” *GAMM-Mitteilungen,* vol. 31, no. 1, pp. 27-58, 2008.

[20] J. Vannoy and J. Xiao, “Real-time adaptive and trajectory optimized manipulator motion planning,” *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 28 – October 2, 2004, Sendai, Japan, pp. 497-502.

[21] N. Sadati and A. Babazadeh, “Optimal control of robot manipulators with a new two-level gradient-based approach,” *Electrical Engineering*, vol. 88, pp. 383-393, 2006.

[22] F. Fahroo, I. M. Ross, “On Discrete-Time Optimality Conditions for Pseudospectral Methods,” AIAA/AAS Astrodynamics Specialist Conference and Exhibit, 21 - 24 August 2006, Keystone, Colorado.

[23] J. Strizzi, I. M. Ross, and F. Fahroo, “Towards real-time computation of optimal controls for nonlinear systems,” AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, CA, 5-8 August 2002.

[24] Z. Bien and J. Lee, “A minimum-time trajectory planning method for two robots,” *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 414-418, June 1992.

[25] J. Peng and S. Akella, “Coordinating multiple robots with kinodynamic constraints along specified paths,” *International Journal of Robotics Research*, vol. 24, no. 4, pp. 295-310, April, 2005.

[26] L. R. Lewis, *Rapid Motion Planning and Autonomous Obstacle Avoidance for Unmanned Vehicles*, Master’s Thesis, Naval Postgraduate School, Monterey, CA, December 2006.

[27] M. A. Hurni, P. Sekhavat, P., and I. M. Ross, “Autonomous Trajectory Planning Using Real-Time Information Updates,” *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 18-21, 2008.

[28] E. G. Gilbert and D. W. Johnson, “Distance Functions and their Application to Robot Path Planning in the Presence of Obstacles.” *IEEE Journal of Robotics and Automation*, vol. 1, No. 1, March 1985, pp. 21-30.

[29] R. R. dos Santos, V. S. Stefffen Jr., and S. de F. P. Saramago, “Robot path planning in a constrained workspace by using optimal control techniques,” *Multibody System Dynamics,* vol. 19, pp. 159-177, 2008.

[30] A. K. Bejczy and R. P. Paul, “Simplified Robot Arm Dynamics for Control,” *Conference on Decision and Control, 20th, and Symposium on Adaptive Processes*, San Diego, CA. December, 1981, pp. 16-18.

[31] J. S. Hesthaven, J. S., Gottlieb, S. and Gottlieb, D., *Spectral Methods for Time-Dependent Problems*, Cambridge University Press, 2007.

[32] Q. Gong, W. Kang, N. Bedrossian, F. Fahroo, F., P. Sekhavat, and K. Bollino, “Pseudospectral Optimal Control for Military and Industrial Applications,” *46th IEEE Conference on Decision and Control*, New Orleans, LA, pp. 4128-4142, Dec. 2007.

[33] I. M. Ross, Q. Gong, and P. Sekhavat, “Low-Thrust, High-Accuracy Trajectory Optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 30, 2007, pp. 921-933.

[34] *Cyton Alpha 7D1G Operations Manual*. Robai, 2008. ftp:///ftp.energid.info/outgoing/cyton/CytonSetup_v1.1.exe.

[35] L. Tsai, *Robot Analysis: The Mechanics of Serial Manipulators and Parallel Manipulators*. New York: John Wiley & Sons, 1999.

[36] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York: John Wiley & Sons, 1989.

[37] M. S. Mahmoud, “Robust Control of Robot Arms including Motor Dynamics,” *International Journal of Control*, vol. 58, no. 4, 1993, pp. 853-873.

[38] Q. Gong, F. Fahroo, and I. M. Ross, “Spectral Algorithm for Pseudospectral Methods in Optimal Control,” *Journal of Guidance, Control, and Dynamics*, vol. 31, no 3, May-June 2008, pp. 460-471.

[39] S. Fortune, G. Wilfong, and C Yap, “Coordinated motion of two robot arms,” in *1986 IEEE International Conference on Robotics and Automation Proceedings,* vol. 3, April 1986, pp. 1216-1223.

[40] K. Sun, and V. Lumelsky, “Path planning among unknown obstacles: The case of a three-dimensional Cartesian arm,” *IEEE Transactions on Robotics and Automation,* vol. 8, no. 6, pp. 776-786, December 1992.

[41] V. J. Lumelsky, “On Fast Computation of Distance Between Line Segments,” *Information Processing Letters 21*, pp. 55-61, August 1985.

[42] D. Sunday, “Distance Between Lines and Segments With Their Closest Point of Approach,” September 2008, http://geometryalgorithms.com/ Archive/algorithm_0106/algorithm_0106.htm.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California